

Georgia State University

ScholarWorks @ Georgia State University

Physics and Astronomy Dissertations

Department of Physics and Astronomy

12-4-2006

Threshold Extension of Gallium Arsenide/Aluminum Gallium Arsenide Terahertz Detectors and Switching in Heterostructures

Mohamed Buhary Rinzan

Follow this and additional works at: https://scholarworks.gsu.edu/phy_astr_diss



Part of the [Astrophysics and Astronomy Commons](#), and the [Physics Commons](#)

Recommended Citation

Rinzan, Mohamed Buhary, "Threshold Extension of Gallium Arsenide/Aluminum Gallium Arsenide Terahertz Detectors and Switching in Heterostructures." Dissertation, Georgia State University, 2006. doi: <https://doi.org/10.57709/1059812>

This Dissertation is brought to you for free and open access by the Department of Physics and Astronomy at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Physics and Astronomy Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

THRESHOLD EXTENSION of GaAs/Al_xGa_{1-x}As TERAHERTZ DETECTORS and SWITCHING in HETEROSTRUCTURES

by

MOHAMED B. RINZAN

Under the direction of Unil Perera

ABSTRACT

In this work, homojunction interfacial workfunction internal photoemission (HIWIP) detectors based on GaAs, and heterojunction interfacial workfunction internal photoemission (HEIWIP) detectors based mainly on the GaAs/Al_xGa_{1-x}As material system are presented. Design principles of HIWIP and HEIWIP detectors, such as free carrier absorption, photocarrier generation, photoemission, and responsivity, are discussed in detail. Results of *p*-type HIWIPs based on GaAs material are presented. Homojunction detectors based on *p*-type GaAs were found to limit their operating wavelength range. This is mainly due to band depletion arising through carrier transitions from the heavy/light hole bands to the split off band. Designing *n*-type GaAs HIWIP detectors is difficult as it is strenuous to control their workfunction.

Heterojunction detectors based on GaAs/Al_xGa_{1-x}As material system will allow tuning their threshold wavelength by adjusting the alloy composition of the Al_xGa_{1-x}As barrier, while keeping a fixed doping density in the emitter. The detectors covered in this work operate from 1 to 128 μm (300 to 2.3 THz). Enhancement of detector response using resonance cavity architecture is demonstrated. Threshold wavelength extension of HEIWIPs by varying the Al composition of the barrier was investigated. The threshold limit of ~ 3.3

THz ($92\ \mu\text{m}$), due to a practical Al fraction limit of ~ 0.005 , can be overcome by replacing GaAs emitters in GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ HEIWIPs with $\text{Al}_x\text{Ga}_{1-x}\text{As}$ emitters. As the initial step, terahertz absorption for $1\ \mu\text{m}$ -thick Be-doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epilayers (with different Al fraction and doping density) grown on GaAs substrates was measured. The absorption probability of the epilayers was derived from these absorption measurements. Based on the terahertz absorption results, an $\text{Al}_x\text{Ga}_{1-x}\text{As}/\text{GaAs}$ HEIWIP detector was designed and the extension of threshold frequency (f_0) to 2.3 THz was successfully demonstrated.

In a different study, switching in GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ heterostructures from a tunneling dominated low conductance branch to a thermal emission dominated high conductance branch was investigated. This bistability leads to neuron-*like* voltage pulses observed in some heterostructure devices. The bias field that initiates the switching was determined from an iterative method that uses feedback information, such as carrier drift velocity and electron temperature, from hot carrier transport. The bias voltage needed to switch the device was found to decrease with the increasing device temperature.

INDEX WORDS: Homojunction, Heterojunction, Terahertz detectors, Free carrier absorption, Reststrahlen band, Interfacial workfunction, Internal photoemission, Infrared detectors, Emitters, Barriers, Dark current, Photo current, Resonance cavity architecture, Responsivity, Quantum Efficiency, Detectivity, BLIP temperature, Threshold wavelength, Negative differential resistance, Tunneling, Switching.

**THRESHOLD EXTENSION of GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ TERAHERTZ
DETECTORS and SWITCHING in HETEROSTRUCTURES**

by

Mohamed B. Rinzan

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree
of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2006

Copyright By

Mohamed Buhary Rinzan

2006

**THRESHOLD EXTENSION of GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ TERAHERTZ
DETECTORS and SWITCHING in HETEROSTRUCTURES**

by

MOHAMED B. RINZAN

Major Professor:	Unil Perera
Committee:	Gennady Cymbaluyk
	Nikolaus Dietz
	Donald Edwards
	Mark Stockman
	Paul Wiita

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Science
Georgia State University
December 2006

To Faraz and Sarah

Acknowledgments

I express my sincere gratitude to the advisor, Unil Perera, for his support, patience, and encouragement throughout my graduate studies. Besides being a mentor, his technical and editorial advice was essential to the completion of this dissertation. He taught me valuable lessons and insights on the workings of academic research in general. I am grateful to: my co-supervisor, Gennady Cymbaluyk, for providing many valuable comments that improved the numerical simulations; Donald Edwards, Director-Brains and Behavior Fellowship program, for offering the Brains and Behavior Fellowship for the academic years 2004/2005 and 2005/2006; and committee members Nikolaus Dietz, Mark Stockman for discussions that helped me understand conceptual issues. I am also grateful to Dr. Paul Wiita for his careful reviewing of the dissertation.

I always loved the good-spirited discussions relating to research with research scientist Steven Matsik, and postdoctoral research scientists Dmitriy Esaev and Hu Zhigao. I would like to thank all the members of the Department of Physics and Astronomy including the Physics Workshop, and my colleagues and friends for their love and help.

Thanks to my loving wife, Faraz, for her love, patience, and understanding during my research, especially during the many nights of coding C++ programs. My mother and my late father receive the deepest gratitude for their love, help, and unconditional support throughout my academic carrier. Finally, Thanks to my little daughter, Sarah, for sacrificing her precious play time with me.

Table of Contents

List of Tables	viii
List of Figures	x
1 Introduction	1
2 Detector Design	6
2.1 Introduction	6
2.2 Interfacial Workfunction	8
2.2.1 HIWIP Detectors	10
2.2.2 HEIWIP Detectors	11
2.3 Optical Field Distribution in a Dielectric Stack	14
2.3.1 Dispersion Characteristics of Intraband Transitions	27
2.3.2 Photocarrier Generation	29
2.4 Internal Photoemission and Hot Carrier Transport	34
2.5 Dark Current and Detector Noise	36
2.6 Responsivity	41
2.7 Detectivity and BLIP Temperature	56
2.8 Response Speed	58
3 Homojunction Interfacial Workfunction Far Infrared Detectors	64
3.1 Introduction	64
3.2 Experimental	65
3.3 Results and Discussion	67
3.3.1 Dark Current	67
3.3.2 Detector Response	68
3.4 Conclusion	87

4	Heterojunction Interfacial Workfunction Far Infrared Detectors	88
4.1	Heterojunction Detectors Operating up to 20 μm	90
4.1.1	Experimental	94
4.1.2	Results and Discussion	94
4.1.3	Conclusion	104
4.2	Effect of Doped Substrate on Resonant Cavity Enhancement	105
4.2.1	Experimental	107
4.2.2	Results and Discussion	109
4.2.3	Conclusion	119
4.3	Threshold Tailorability in Heterojunction Terahertz Detectors	120
4.3.1	Experimental	128
4.3.2	Results and Discussion	128
4.3.3	Conclusion	131
4.4	Free Carrier Absorption in $\text{Al}_x\text{Ga}_{1-x}\text{As}$ Epitaxial Films	132
4.4.1	Reflectance and Transmittance Measurements	132
4.4.2	Results and Discussion	134
4.4.3	Conclusion	140
4.5	Threshold Extension Using $\text{Al}_x\text{Ga}_{1-x}$ Emitters	144
4.5.1	Experimental	145
4.5.2	Results and Discussion	145
4.5.3	Conclusion	154
5	Hot Electron GaAs/$\text{Al}_x\text{Ga}_{1-x}\text{As}$ Heterostructure Design for Pulsing	155
5.1	Introduction	155
5.2	Switching in Heterostructure Hot Electron Devices	157
5.3	Distribution of the Emitted Current Density	158
5.3.1	Tunneling Probability of the Carriers	161
5.3.2	Switching Observed in a Si-Homojunction Device	173
5.4	Conclusion	173

APPENDICES

A	Programs Used for Response Modeling	176
A.1	Class Definitions and Their Implementations	177
A.2	Optical Constants Library	305
A.3	Digital Filtering of Interference Fringes	306
B	Programs Used for Switching in Heterostructures	308
	Bibliography	xxiii
	Acronyms	xxx

List of Tables

2.1	The structure parameters of detectors considered to demonstrate the effects of vital parameters on the detector response. The last two listed have already been tested, and are used here for the discussion's continuity. The top and bottom contacts of the structures are highly doped (denoted by the superscripts ++) GaAs, and the barriers are undoped $\text{Al}_x\text{Ga}_{1-x}\text{As}$. The substrate is either highly doped or SI-GaAs. Notations W , N , and λ_0 denote the layer thickness, emitter/barrier units and the threshold wavelength of the detectors, respectively.	42
3.1	Main parameters of the three device structures as confirmed by secondary ion mass spectroscopy (SIMS). Labels N_{tc} (W_{tc}), N_{em} (W_{em}), N_b (W_b), and N_{bc} (W_{bc}) denote the doping density (thickness) of the top contact, emitter, barrier, and the bottom contact layers of the structures, respectively. After processing the devices, the etched-down thickness of the emitters is $\sim 800 \text{ \AA}$	66
3.2	Figures of merit (peak responsivity, quantum efficiency and specific detectivity) for the single emitter detectors RU001, RU002 and RU003 at $\lambda = 34 \mu\text{m}$ compared to the multi-emitter detector 9604. The thickness of the emitter (800 \AA) and the bottom contact ($1 \mu\text{m}$) for all three single emitter detectors are the same.	78
3.3	Figures of merit (peak responsivity, quantum efficiency and detectivity) for the single emitter detectors RU001, RU002, and RU003 at $\lambda = 58$ and $64 \mu\text{m}$. These wavelengths correspond to transition of carriers from the ground to the third and to the second excited states of the impurity-well (carbon), respectively. The maximum responsivity is achieved for detector RU003, whereas high density of defects would not allow the electric field to be increased significantly for the detector RU001, and detector RU002 has the lowest barrier thickness resulting in the lowest impurity density.	78

4.1	Parameters of the detectors used in the measurements. In all cases, the emitters were 188 Å-thick GaAs, the barriers were 1250 Å-thick $\text{Al}_{0.12}\text{Ga}_{0.88}\text{As}$, and the contacts were doped to $1 \times 10^{19} \text{ cm}^{-3}$. For all three detectors, $\sim 900 \text{ Å}$ (assuming a depletion of $\sim 100 \text{ Å}$) of the top contact layer was left after etching to form the first emitter. Also shown are the measured values of responsivity and D^*	100
4.2	Device parameters of two detectors (labeled HE0205 and 1329) used to demonstrate the resonance cavity effect. These detectors have the same parameters, except for HE0205 being p -type and grown on a semi-insulating (SI)-substrate whereas 1329 is n -type and grown on a n -type substrate, with emitters doped to $\sim 5 \times 10^{18} \text{ cm}^{-3}$. Both detectors have 12 emitter/barrier units and the thickness of emitters and barriers are 188 Å and 1250 Å, respectively.	108
4.3	Predicted and measured barrier heights, threshold frequencies and wavelengths from the model, Arrhenius plots, and the spectra for the three detectors showing the threshold wavelength variation with the Al composition in the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barriers. The small variation between the model and spectral values is probably due to deviations of the actual parameters from the design values.	130
4.4	Measured parameters of the p -type $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial films. The 1 μm -thick Be:doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ films were grown on 520 μm -thick SI-GaAs (100) substrates. The doping density (N_p), Al composition, and the epitaxial film thickness were obtained from SIMS. The plasma frequency (ω_p) and the free carrier damping constant (ω_0) were obtained from fitting the reflectance spectra to the model.	133
4.5	Variation of threshold frequency (wavelength) with the applied bias field for detector V0207. The increasing field decreases the band bending in the barrier which decreases the threshold frequency.	151

List of Figures

- 2.1 (a) Schematic of a single emitter HEIWIP detector after processing. Dielectric layers n^+ -GaAs, i -AlGaAs, and n^+ -GaAs serve as the emitter, barrier and bottom contact, respectively. Here, the emitter is also serving as the top contact layer. (b) Partial energy band diagram showing the interfacial workfunction, and internal photoemission of a carrier. The downward arrow shows the direction of photocarrier drift due to applied bias. 9
- 2.2 (a) Band diagram showing the band gap narrowing and the Fermi level shift in the emitter of a n -type detector. The upper dashed and solid curves correspond to the conduction band of barrier and emitter, respectively. The slanted arrow in the conduction band of the emitter shows an intraband transition. The photoexcitation involves a phonon to conserve momentum. (b) Variation of the conduction band edge and the Fermi level (at $k = 0$) along the growth direction, showing the interfacial workfunction barrier. The vertical arrow corresponds to the photoexcitation shown by the slanted arrow on the left figure. 12
- 2.3 (a) Variations of the major band edge due to majority carrier interaction and light-heavy hole interaction in p -type GaAs. Also shown is the Fermi level shift including the effects of multiple bands. The change in the workfunction over a doping density change from 1×10^{18} to $1 \times 10^{19} \text{ cm}^{-3}$ is $\simeq 1 \text{ meV}$. (b) Variations of the major band edge due to majority carrier interaction and anisotropy in the conduction band in n -type GaAs. The shift in Fermi level in n -type GaAs is very large due to low electron *density of state mass*. The workfunction vanishes around a doping density of $\sim 1.5 \times 10^{17} \text{ cm}^{-3}$ which is the Mott's critical density for a n -type GaAs. The contribution to bandgap narrowing from the anisotropy in the conduction band is also shown. . . . 13

- 2.4 (a) A dielectric stack and the electric (E) field components within its layers. Subscripts denote the layer index. For a given dielectric layer, the permittivity, permeability, complex refractive index, and thickness are given by ϵ , μ , \tilde{n} and t , respectively. The incident and the emerging media are labeled as a and b , and θ denotes the incident angle. (b) Magnetic (H) field amplitudes for s -polarized plane waves at the $i/(i+1)$ interface. 15
- 2.5 Amplitude variation of the reflection (ρ) and transmission (τ) coefficients of (a) electric field and (b) magnetic field with the incident angle. The signs \perp and \parallel refers to s - and p -polarized light, respectively. The light is incident from the vacuum side to a $5 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs epitaxial film; φ_B is the Brewster's angle for the vacuum/film interface. 18
- 2.6 Phase variation of the reflection (ρ) and transmission (τ) coefficients of (a) electric field and (b) magnetic field with the incident angle. The light is incident from the vacuum side to a $5 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs epitaxial film: φ_B is the Brewster's angle for the vacuum/film interface; $\varphi_\rho = 0$ implies that the reflected wave is in phase with the incident wave, and $\varphi_\tau = 0$ implies that transmitted wave is in phase with the incident wave. 19
- 2.7 Amplitude variation of the reflection (ρ) and transmission (τ) coefficients of (a) electric field and (b) magnetic field with the incident angle. The signs \perp and \parallel refers to s - and p -polarized light, respectively. Radiation penetrates from $5 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs epitaxial film to the vacuum. The Brewster's angle, φ_B , and the angle for total internal reflection for the film/vacuum interface are both $\sim 11^\circ$ 20
- 2.8 Variation of (a), (b) Reflectance and (c), (d) Transmittance spectra with incident angle for s -polarized (TE-Mode) and p -polarized (TM-Mode) light incident on a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer. The absorption of the dielectric layer has not been considered in this. The dip at 272.6 cm^{-1} is due to transverse optical (TO) phonons in the GaAs layer, and the plasma frequency is 411.3 cm^{-1} 23
- 2.9 (a) Reflectance and (b) Transmittance of s -polarized (TE-Mode) and p -polarized (TM-Mode) light for $\lambda = 50 \text{ }\mu\text{m}$. Radiation is incident on a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer from the ambient (vacuum) side. The p -polarized reflectance and transmittance reaches the minimum and the maximum, respectively, at the Brewster's angle ($\theta = 76.3^\circ$) for the vacuum/GaAs layer interface. 24

- 2.10 Variation of (a), (b) Reflectance and (c), (d) Transmittance spectra with incident angle for s -polarized (TE-Mode) and p -polarized (TM-Mode) light when infrared light is incident on ambient (here vacuum) from a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer. The dip at 272.6 cm^{-1} is due to transverse optical (TO) phonons in the GaAs layer, and the plasma frequency is 411.3 cm^{-1} . The large gap between incident angles $\theta = 0$ and 15° is due to reflection enhancement from total internal reflection of light. 25
- 2.11 Reflectance and transmittance of s -polarized (TE-Mode) and p -polarized (TM-Mode) light for $\lambda = 50 \text{ }\mu\text{m}$. Radiation is incident on ambient (vacuum) from a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer side. The p -polarized reflectance and transmittance reaches the minimum and the maximum, respectively, at the Brewster's angle ($\theta = 10.6^\circ$) for the GaAs layer/ambient(vacuum) interface. The total-internal-reflection angle is almost the same as the Brewster's angle for the chosen doping density. 26
- 2.12 Variation of skin depth with doping density for (a) p - and (b) n -type GaAs dielectric layers. Lower skin depth for n -type is due to lower effective mass of electrons compared to holes. The dip around $37 \text{ }\mu\text{m}$ is due to GaAs TO-phonons. 30
- 2.13 Comparison of photocarrier generation rate between the layers in (a) n -type and (b) p -type GaAs/ $\text{Al}_{0.10}\text{Ga}_{0.90}\text{As}$ detectors. Each detector has a single emitter with a doping density of $1 \times 10^{18} \text{ cm}^{-3}$. The rate increase in the n -type detector is the result of its better photoabsorption probability due to lower electron effective mass. The generation in the top contributes to photocurrent when the carrier injection is from the top, and that of the bottom contributes when it is from the bottom contact layer. 32
- 2.14 Comparison of internal photoemission efficiency, η_i , between n - and p -type GaAs emitters with a doping density of $1 \times 10^{18} \text{ cm}^{-3}$. Interfacial workfunction of the barrier is set to $\Delta = 17.7 \text{ meV}$ to obtain a threshold of $\lambda_0 = 60 \text{ }\mu\text{m}$. Photoemission efficiency is better in p -type GaAs emitters due to higher effective mass of its carriers. The efficiency decreases with the increasing emitter thickness, and is also a function of the scattering lengths. Generally, the energy independent values of $L_e = 400 \text{ nm}$ and $L_p = 20 \text{ nm}$ are used for doped GaAs films. 37
- 2.15 Experimental and model bias dependence of the dark current for detector HE0204 at temperatures where the thermionic component becomes dominant. The interfacial workfunction, $\Delta=77 \text{ meV}$, used in the model was derived from the Arrhenius plot for the detector. The deviation at high bias is due to the increase in the tunneling probability. 39

- 2.16 (a) Photon absorption as a function of bottom contact thickness for a single barrier n -type GaAs/Al_{0.26}Ga_{0.74}As detector (structure I). Doping density of the contacts are $1 \times 10^{19} \text{ cm}^{-3}$. The absorption around the short wavelength region increases as the thickness (W_{bot}) of the bottom contact is increased. The ripples in the spectra are due to Fabry-Pérot interference in the bottom contact layer. (b) Detector response for $W_{\text{bot}} = 0.7 \text{ }\mu\text{m}$. The left and right response peaks correspond to the third and the first order resonance, respectively. The threshold wavelength is $\lambda_0 = 27 \text{ }\mu\text{m}$. The inset shows the insignificant increase in the reverse bias response over a large increase in the bottom contact thickness: from $W_b = 0.7$ to $10 \text{ }\mu\text{m}$ (a factor of ~ 140). (c) Detector response for $W_{\text{bar}} = 0.2 \text{ }\mu\text{m}$. The reduced epitaxial stack thickness decreases the cavity length, which results in the redistribution of the resonance peaks. 44
- 2.17 (a) Photocarrier generation rate for a single barrier detector (structure II) with a n -type top contact and an undoped barrier grown on a n -type substrate. The doping densities of the top contact and substrate are both $2 \times 10^{18} \text{ cm}^{-3}$, and the Al_{0.11}Ga_{0.89}As barrier thickness is $1 \text{ }\mu\text{m}$. Here, the substrate serves as the bottom contact. The arrows indicate the GaAs-like and AlAs-like TO-phonon effects. The generation of photocarriers in the substrate is within three skin depth scales for a given wavelength. (b) Total carrier generation as a function of top contact thickness. The carrier generation around the designed peak slowly increases with the contact thickness. (c) Responsivity spectra for $W_{\text{top}} = 0.2 \text{ }\mu\text{m}$ under reverse bias. The carrier injection is from the top contact. The interfacial work function corresponds to $\lambda_0 \sim 27 \text{ }\mu\text{m}$ 46
- 2.18 Calculated absorption probability in the emitters of structure III. The structure consists of a 400 nm-thick p -type top contact and 700 nm-thick p -type bottom contact of doping density $1 \times 10^{19} \text{ cm}^{-3}$. This has nine emitter/barrier units. The 70 nm-thick emitters are p -type with a doping density of $3 \times 10^{18} \text{ cm}^{-3}$, and the 100 nm-thick barriers are undoped. The substrate is n -type with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$ 48
- 2.19 Calculated responsivity spectra for structure III with nine emitter/barrier units. The structure consists of p -type top and bottom contacts with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$, and a thickness of 400 and 700 nm, respectively. The 70 nm-thick emitters are p -type with a doping density of $3 \times 10^{18} \text{ cm}^{-3}$, and the 100 nm-thick barriers are undoped. The n -type substrate has a doping density of $1 \times 10^{19} \text{ cm}^{-3}$. The forward and reverse spectra represent the carriers photoinjected from the top and bottom contacts, respectively, with all the emitters contributing to both bias directions. 49

- 2.20 Calculated temperature dependence of dark current in structure III. The applied bias is 2 kV/cm, and the interfacial workfunction corresponds to $\lambda_0 = 15 \mu\text{m}$. Photocurrent levels of 8.5 and 21 μA correspond to 300 K background illumination at field of view (FOV) = 60° and 180° , respectively. The quantum efficiency (η) and gain are assumed to be unity. The Intersections of the curves show that $T_{\text{BLIP}} = 75$ and 82 K for the respective FOVs. 51
- 2.21 (a) Calculated total absorption in the top and bottom contacts, and in the emitters for structure IV with 22 emitter/barrier units. The absorption in the emitters contribute to both the forward and reverse directions. (b) The responsivity spectra for the same structure as a function of emitter/barrier units. The structure consists of a p -type top contact and a n -type bottom contact with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$. Emitters are p -type GaAs with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$. The threshold wavelength, $\lambda_0 = 35 \mu\text{m}$. 52
- 2.22 (a) Experimental and model responsivity spectra of detector 2409 with 30 emitter/barrier units. Doping density in the top and bottom contacts is $2.4 \times 10^{19} \text{ cm}^{-3}$, and the emitter has a doping density of $2 \times 10^{18} \text{ cm}^{-3}$. Thickness of the top contact, barrier layer, emitter and bottom contact are 208, 77, 15 and 730 nm, respectively. Total thickness of the structure is $3.5 \mu\text{m}$. The responsivity of an optimized 2409 structure is shown in (b). The responsivity increases by 1.7, 2.1 and 4.5 times if an additional n -type buffer layer, or a n -type bottom contact, or an increased emitter doping density is used, respectively. 54
- 2.23 Experimental and model BLIP temperature vs. bias for detector HE0204. The background temperature was 300 K and the FOV of the detector at the cold stop was 62° 57
- 2.24 The solid line shows the model detectivity spectra for the 2409 HEIWIP detector in the BLIP regime (λ_0 is $70 \mu\text{m}$). Background temperature is $T_{\text{BG}} = 300 \text{ K}$ and FOV = 180° . The BLIP temperature, $T_{\text{BLIP}} = 13 \text{ K}$. Model detectivities for structures similar to 2409, but with a $1.5 \mu\text{m}$ -thick n -type buffer layer doped to $1 \times 10^{19} \text{ cm}^{-3}$, or with a n -type bottom contact, or with the emitter doping density increased to $1 \times 10^{19} \text{ cm}^{-3}$ are shown for comparison. The top solid line represents the detectivity of an ideal detector with the same λ_0 59
- 2.25 Bias dependence of responsivity in 2409 HEIWIP FIR detector with 30 emitter/barrier units; $\lambda_0 = 70 \mu\text{m}$ 60

- 3.1 Schematic diagram of the p -type GaAs single emitter HIWIP detector after processing. A window is opened on the top for front illumination. The three structures RU001, RU002 and RU003 mainly differ in their barrier thickness, which is 4, 0.1 and 1 μm , respectively. For all three detectors, the post-etched emitter thickness is ~ 800 Å, although the voltage drop still would apply through the unetched length of the device. The etching will increase the energy throughput to the emitter layer of the detector. 66
- 3.2 (a) Dark current curves for the three detectors at 4.2 K. The asymmetry in the dark current curves is due to non-uniformity in the structures. The rapid rise of dark current in RU001 can be attributed to defects in the barrier. The zoomed portion shown in the inset clearly shows the deviations. (b) Arrhenius plots under forward bias fields of 0.05, 3, and 1 kV/cm for detectors RU001, RU002 and RU003, respectively. (c) The Arrhenius plot for RU002 shows different interfacial workfunction in the forward and reverse direction. This is due to barrier lowering arising from asymmetry. 69
- 3.3 Variations of workfunction obtained through Arrhenious plots with the applied field. Detector RU001, with the 4 μm -thick barrier layer, has an almost constant workfunction up to 0.1 kV/cm, decreasing sharply afterwards. Whereas, the other two show the expected bias variation due to image force lowering. The deviation in RU001 explains the sharp increase observed in its dark current (see Fig. 3.2). 70
- 3.4 The experimental (solid line) and model (dashed line) absorption spectra for the structures with different barrier thickness at room temperature. The absorption measurements were done for pieces without etched top contacts. RU001 has a 4 μm -thick barrier layer whereas RU002 and RU003 have 0.1 and 1 μm -thick barriers, respectively. The first order cavity peak for structure RU001 was expected around the position, $\lambda = 68$ μm , shown by the arrow. Higher order peaks for structure RU001, and peaks of all orders for structures RU002 and RU003 fall outside the measured spectral range. 72
- 3.5 The experimental responsivity spectra for detectors at 4.2 K under different forward bias. The maximum responsivity observed was at $\lambda = 34$ μm , and it is 3.3, 1.4, and 4 A/W for detectors RU001, RU002 and RU003, respectively. The first order cavity peak for detector RU001 is around the threshold wavelength. The sharp drop around 37 μm is due to the high reflection in the reststrahlen band. Arrows 1 and 2 indicate the transitions between the ground and the excited impurity (carbon) states. 74

3.6	Model fit to experimental spectra of detectors RU001, RU002, and RU003 under forward bias at $T = 4.2$ K. The dip in the responsivity around $37 \mu\text{m}$ is due to high reflection in the reststrahlen band of GaAs. Unlike in other fits, the model deviates from the data significantly around the region where contribution to photoionization from the impurity atoms in the barrier is significant. The model used here does not include this mechanism.	79
3.7	Responsivity vs. Wavelength for reverse bias at $T = 4.2$ K; solid lines: experimental data, dashed lines: model. The matching of model to data was done at electric fields 0.05, 0.5, and 1.5 kV/cm for detectors RU001, RU002 and RU003, respectively. The sharp dip in responsivity around $\sim 37 \mu\text{m}$ is due to strong interaction of photons with phonons in GaAs.	81
3.8	Calculated mean square of the optical electric field across the emitter and bottom contact layers of the three detectors. The lowest electric field produced the weakest responsivity for detector RU002. The sharp drop in the vicinity of GaAs-TO phonons ($36.7 \mu\text{m}$) is due to strong TO-phonon photon interaction. The peak electric field at LO phonon frequency ($33.9 \mu\text{m}$) is due to the dropping of the permittivity to zero. The drop and rise of electric field result in the drop and rise of the responsivity as in Figs. 3.5 and 3.9.	83
3.9	The experimental responsivity spectra for detectors at $T = 4.2$ K under reverse bias. Maximum responsivity, excluding the response due to impurity photoionization of carriers in the barrier, is at $\lambda = 34 \mu\text{m}$, and its values are 3.5, 3.6 and 7.4 A/W for detectors RU001, RU002 and RU003, respectively. The first order cavity peak for detector RU001 is around the threshold wavelength. The sharp drop around $37 \mu\text{m}$ is due to the high reflection in the proximity of reststrahlen band. Arrows 1 and 2 indicate the transitions of holes between ground and excited states of the coulomb well formed by impurity (carbon) atoms in the barrier.	84
3.10	Calculated optical electric field distribution of the standing wave generated within the dielectric stacks for $\lambda = 34 \mu\text{m}$. The maximum field is on the surface of the emitter. The low field strength for detector RU002 is due to its stack-thickness being the lowest compared to RU001 and RU003.	85
3.11	Calculated hot carrier generation rate across the structures for $\lambda = 34 \mu\text{m}$. High generation rate in the emitter and the bottom contact layers is due to their high doping density. For detector RU002, the carrier generation rate of the emitter layer is lower than that of the bottom contact layer.	86

- 4.1 Band diagram showing the band gap narrowing and the Fermi level shift in a p -type doped GaAs emitter, and the band offset of the adjacent $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier. This is a type-I system where the bandgap of GaAs is completely within the band gap of $\text{Al}_x\text{Ga}_{1-x}\text{As}$. The conduction band to valance band offset ratio is $\sim 65:35$ of the total offset. Therefore, at $T = 4.2$ K, the valance band offset varies as $530 \times x$, where x is the Al composition in the barrier. In p -type GaAs, the offset due to doping only varies by ~ 1 meV for a doping density change from 1×10^{18} to $1 \times 10^{19} \text{ cm}^{-3}$ (see Fig. 2.3(a)). The interfacial workfunction (Δ) is the sum of the doping offset (Δ_d) of the emitter and the valance band offset (Δ_x) of the barrier. The dashed line on the flat band diagram on right shows the valance band edge ($k = 0$) of the barrier before offset ($x = 0$), resembling the interface in a HIWIP detector. 91
- 4.2 Band diagram showing the band gap narrowing and the Fermi level shift in a n -type doped GaAs emitter, and the band offset of the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier. Unlike in p -GaAs, the doping offset in n -type strongly depends on doping density. For a doping density of $1 \times 10^{19} \text{ cm}^{-3}$, the offset results in the Fermi level being above the barrier at $x = 0$. The interfacial workfunction $\Delta = \Delta_x + \Delta_d$, where $\Delta_d < 0$ for $n > 1 \times 10^{17} \text{ cm}^{-3}$. This negative offset should be taken into account in determining the alloy fraction of the barrier layer for a given interfacial workfunction. The dashed line on the flat band diagram on the right shows the valance band edge ($k = 0$) of the barrier before offset ($x = 0$). 92
- 4.3 Threshold wavelength vs. Al fraction for p and n -type GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ HEIWIP detectors. The threshold wavelength for n -type detectors increases rapidly with decreasing alloy fraction. For p -type, the Al fraction required to increase λ_0 close to $100 \mu\text{m}$ is less than 1 % whereas for n -type it is ~ 3.5 %, which is relatively easy to control. The upper limit in Al fraction corresponds to a transition point beyond where the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier becomes indirect and photoexcited carriers need an additional phonon for emission. Meanwhile, for practical purposes, the lower limit is $x \simeq 0.005$. Aluminum fraction growth accuracy and the transition from alloy to iso-electronic doping behavior in MBE grown structures define the lower limit of x in HEIWIP detectors. 93
- 4.4 (a) Dark current vs. Bias at different device temperatures illustrating the thermionic nature of the current. Also shown is the 300 K background photocurrent obtained at 40 K (solid line). The temperature $T = 4.2$ K gives a maximum bias of 1.5 V for BLIP operation. (b) A modified Arrhenius plot giving a threshold wavelength of $\lambda_0 \simeq 20 \mu\text{m}$. (c) Plot of BLIP temperature vs. Maximum bias voltage. The region under the curve gives the combinations of bias voltage and device temperature that allow the detector to operate within BLIP condition. 97

- 4.5 (a) Dark current at $T = 77$ K for the three detectors. The doping density in the emitters of HE0204, HE0205 and HE0206 are 1×10^{18} , 3×10^{17} and $1 \times 10^{17} \text{ cm}^{-3}$, respectively. Similar variation is observed at $T = 4.2$ K where the current is much lower. (b) Comparison of dark current density between mesas of HE0206 with different electrical areas. The measurements were done at $T = 77$ K. The four curves appear essentially as a single curve due to high uniformity. 98
- 4.6 (a) Measured responsivity of HE0204 at different temperatures. The peak response was 100 mA/W at $\sim 12.5 \mu\text{m}$. The response remained nearly constant up to 40 K and then decreased rapidly. This is consistent with the BLIP temperature of 40 K estimated from dark and background currents. (b) Responsivity for HE0205 at 4 V bias and $T = 4.2$ K, showing the greatly reduced response observed when doping is reduced. 99
- 4.7 (a) Model absorptance in the first emitter (200 Å-thick remainder of the top contact) for devices with n - and p -type bottom contacts. Both designed with the same parameters as of HE0204, but with reduced emitter/barrier units to have the first order resonance peak near $15 \mu\text{m}$. The use of n -type material greatly increases the absorption in the 10–20 μm range. (b) Model response for the two designs. The use of n -type material leads to almost a factor of 4 increase in the peak response. 102
- 4.8 Variation in the experimental workfunction with the bias for detector 1329 with a doped substrate. The workfunction drops rapidly in the forward bias, and varies slowly for reverse bias. The inset shows a band diagram of the first barrier with the contact on the left and an emitter on the right. The higher doping in the contact gives a higher Fermi level. Under zero bias (the top diagram), the barrier slopes up with the highest value adjacent to the emitter. Under low bias (the middle diagram), the barrier slopes up with the effective barrier height Δ reduced by the applied bias. Under high bias (the bottom inset), the barrier slopes down. Here, Δ is determined from the contact and has only a slow variation with bias. 111
- 4.9 Response under forward bias for detector HE0205 at $T = 4.2$ K. Response was not observed under reverse bias. The dashed line shows the modeled response for the detector parameters of HE0205. The arrow at $\lambda \simeq 3 \mu\text{m}$ indicates the response due to carriers photoexcited from heavy-hole (HH) and light-hole (LH) bands to split-off (SO) band of GaAs. This mechanism is not included in the model. 112

- 4.10 (a) Response under forward and reverse bias for detector 1329 (grown on a n -type substrate) at $T = 4.2$ K. The response is strong under forward bias. The peak at $10\ \mu\text{m}$ is enhanced by waveguide effects under the contact region. The dashed line shows the modeled response for the same detector parameters. (b) Comparison of response for structures grown on doped and SI-GaAs substrates. As expected, the numbered peaks correspond to minima of the reflectance spectra shown in Fig. 4.11. 113
- 4.11 Reflectance measurements for detector structures (a) HE0205 and (b) 1329. The measurements show an increased reflection from detector structure 1329 as well as the reduced reflection (labeled 1–4) corresponding to the response peaks labeled in Fig. 4.10. 116
- 4.12 The predicted difference in response for devices with n -type emitters grown on a n -substrate, n -type emitters grown on a SI-substrate, and p -type emitters grown on a SI-substrate. Other parameters are the same as for HE0205. The change from p - to n -type emitters produces only a minimal change whereas the use of a n -type substrate produces an increase of ~ 8 times in the detector response. 117
- 4.13 (a) Response for an optimized device with 32 periods of $200\ \text{\AA}$ -thick p -type GaAs emitters doped to $3 \times 10^{18}\ \text{cm}^{-3}$ and $350\ \text{\AA}$ -thick undoped $\text{Al}_{0.15}\text{Ga}_{0.85}\text{As}$ barriers grown on a n -type GaAs substrate of doping density $5 \times 10^{18}\ \text{cm}^{-3}$. The peak responsivity has been increased to near $4\ \text{A/W}$. (b) The calculated D^* with a peak value of $\sim 2 \times 10^{10}\ \text{cm}^2/\text{V}\cdot\text{s}$ as well as the calculated D^* for an ideal detector. 118
- 4.14 Band diagram of the emitter/barrier interface for a device using doped AlGaAs as the emitter and GaAs as the barrier to extend f_0 beyond $2.7\ \text{THz}$. The parameters shown are for a device with $f_0 = 0.9\ \text{THz}$ ($\lambda_0 \sim 335\ \mu\text{m}$). The dashed line in the emitter indicates the Fermi level location if the emitter was GaAs. The contributions to Δ from the doping (Δ_d) and Al fraction (Δ_{Al}) are indicated by the vertical arrows. The effective barrier is $\Delta = \Delta_{\text{Al}} - \Delta_d$. 121
- 4.15 (a) A partial band diagram of the top two periods of a HEIWIP detector with doped GaAs as the emitter layer and undoped AlGaAs as the barrier. The effective width of the nonzero field region is indicated by δ . The emitter doping forms a 3-D carrier distribution. The detectors have $3 \times 10^{18}\ \text{cm}^{-3}$ Be-doped $158\ \text{\AA}$ -thick emitters and $800\ \text{\AA}$ -thick $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barriers. (b) Schematic diagram of the detectors after processing. A window is opened on the top for front illumination. 122

- 4.16 Experimental responsivity spectra for detectors 2409, 2410 and 2411 at 3.5 kV/cm obtained at $T = 4.2$ K. The only difference between the detectors is the Al fraction of the barriers, which is $x = 0.02$, 0.01 and 0.005 for 2409, 2410 and 2411, respectively. The data show a decrease in f_0 with decreasing x . The sharp dip near $f = 8$ THz is due to strong interaction between GaAs-like TO-phonons and photons which results in a strong reflection. The small dip at $f = 10.8$ THz is due to AlAs-like TO-phonons in the barrier layers. The arrows indicate both the threshold frequencies of the detectors (x -axis) and the measurement noise levels (y -axis). The different levels are due to dynamic resistance differences of the detectors. The threshold variation; $\lambda_0 = 65$, 84 and 92 μm for detectors 2409, 2410 and 2411, respectively, can be seen clearly. 126
- 4.17 (a) Model spectra for 150 Å-thick, $3 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs emitter with $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier forming a single layer detector. Al fractions correspond to the experimental detectors. The model spectra are scaled to match the bias field used for the experimental spectra. (b) The variation in threshold frequency with Al fraction showing a comparison of the spectral threshold, the threshold predicted from the Arrhenius plot, and the model result. The discrepancy between the experimental and predicted results is probably due to small variations in Al fraction. 127
- 4.18 Experimental (solid line) and model (dashed line) reflectance spectra for the Be:doped epitaxial films grown on 520 μm -thick GaAs SI-substrates. The sharp peaks at ~ 37 μm are due to the interaction of radiation with GaAs-like TO phonons, and the arrows at ~ 28 μm point to small peaks due to interaction with AlAs-like TO phonons. The strength of AlAs-like phonons increases with Al composition as shown. The ripples towards the FIR matches the Fabry-Pérot interference in the SI-substrate. 137
- 4.19 (a) Experimental absorption coefficient (α) in the range 10 – 100 μm for Be:doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ MBE-grown epitaxial films. the measurements were done at room temperature. The dashed, solid and dotted curves show α for films with doping density 3×10^{18} , 4.7×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$, respectively. The region shown by the break corresponds to the combined reststrahlen band of GaAs-like and AlAs-like TO-phonons of $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$. (b) The absorption coefficient is almost independent of wavelength in the FIR range 100 – 400 μm . The absorption coefficient in this region are 3×10^3 , 3.5×10^3 and $5 \times 10^3 \text{ cm}^{-1}$ for $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films with doping density 3×10^{18} , 4.7×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$, respectively. 139

- 4.20 (a) Experimental absorption coefficient (α) in the range 100–400 μm for Be:doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epilayers with different Al composition. The curve was derived from room temperature reflection and transmission spectra. Free carrier absorption is found to be almost independent of wavelength in the 100–400 μm range. The value of α has decreased from $\sim 3.5 \times 10^3$ for $x = 0.01$ to $\sim 3 \times 10^3 \text{ cm}^{-1}$ for $x = 0.16$. (b) Sub-linear relationship of the free carrier absorption coefficient with acceptor (Be) doping density, $\alpha \propto p^{0.5}$, for $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films. The values represent the average in the range 100–200 μm 141
- 4.21 The model and experimental free hole absorption coefficient for a $7.1 \times 10^{18} \text{ cm}^{-3}$ Be:doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epilayer from 2 to 400 μm at room temperature. The peak around 3 μm is due to carrier transitions from the heavy and light hole bands to the split off band of $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$. The model does not include this mechanism. 142
- 4.22 (a) A partial band diagram of the active region of the HEIWIP device, with an intrinsic barrier under bias showing the contributions to the workfunction from the band gap narrowing (Δ_d) in the doped emitter and the $\text{Al}_x\text{Ga}_{1-x}\text{As}/\text{GaAs}$ valance band offset ($\Delta_x < 0$). The dashed line indicates where the location of the valance band edge in the barrier would be if it were GaAs. Here, $\Delta = \Delta_d + \Delta_x$ where x is the Al fraction. (b) Band bending due to residual doping in the barrier. For zero bias field Δ_b is $\sim 13 \text{ meV}$. Because Δ_b is a function of the bias field, for a given bias $\Delta = \Delta_d + \Delta_x + \Delta_b$ and Δ decreases with the bias. 147
- 4.23 Variation of the workfunction, Δ , with the bias field for three mesas with different electrical areas. The workfunctions at different bias fields were obtained using Arrhenius plots. The zero bias workfunction is $\sim 17 \text{ meV}$ for all the mesas. Inset shows the experimental and model variation of barrier height for the device with $1000 \times 1000 \mu\text{m}^2$ electrical area. The variation with the bias is due to band bending caused by space charge in the barrier layer. 149
- 4.24 The variation of responsivity with applied field for detector V0207 at $T = 4.8 \text{ K}$. The peak responsivity, 9 A/W at 9.6 THz, was obtained at 1.5 kV/cm. The increase in response with the field around f_0 is due to threshold shift with the bias. The sharp dip at $\sim 8 \text{ THz}$ is due to the interaction of radiation with GaAs-like TO phonons. The bias field decreases the effective work function pushing f_0 towards 2 THz with the increasing field. 151

4.25	Calculated responsivity spectra for structures with similar parameters as in V0207 and emitter/barrier units of $N=10, 40$, and 50 are shown. The responsivities are for a bias field of ~ 2.0 kV/cm. The detector V0207 is not cavity optimized, whereas the model structures with $N=40$ and 50 are optimized for 5.6 and 4.9 THz, respectively. The cavity peaks of order $3, 7, 9\dots$ are seen, whereas order 5 (7.7 THz) falls within the reststrahlen band of GaAs. The inset shows the experimental responsivity of detector V0207 for a bias field of 2 kV/cm with the calculated responsivity with an optical gain of 2 . The arrows indicate the cavity peaks of the detector. The shoulder at $14\text{ }\mu\text{m}$ on both the experimental and the model curves corresponds to the third order cavity peak with the first order (7.3 THz) peak falling within the reststrahlen band of GaAs.	153
5.1	Partial band diagram of the GaAs/ $\text{Al}_{0.46}\text{Ga}_{0.54}\text{As}$ heterostructure hot-electron device (a) at thermal equilibrium, and (b) under reverse bias. Width of the space charge layer where the carriers trap to form a 2D-electron gas is denoted by δ . The diagram is for high bias where the thermionic emission is the predominant current mechanism. Fields due to external bias are denoted by F_1 and F_2 for drift and barrier layers, respectively.	159
5.2	Conduction band edge of hetero-structure hot electron device (HHED) under bias. Subscript Γ refers to the Γ -valley of the conduction band. The tunneling current components from the bulk electrons in the drift layer ($j_{T\Gamma}(k_z)$) and the bound electrons in the V -shaped well in the space charge region ($j_{TB}(k_z)$) are shown by arrows. The band bending in the barrier due to space charge is negligible and the field in the barrier is assumed to be constant. The label Γ represents the electrons in that valley.	162
5.3	Current density vs. Electron temperature in the drift layer of an HHED. The electron temperature increases rapidly after a certain current signaling the switching from the low conductance to the high conductance branch. . . .	166
5.4	Energy distribution of the emitted current density, corresponding to Γ -valley electrons, for six different values of the temperature. The solid and dashed lines correspond to bias values of 1.0 and 1.2 V, respectively. The dotted line shows the position of the barrier. As the temperature increases the distribution shifts above the interfacial barrier.	168
5.5	Variation of the current density with lattice temperature. The label “switching point” indicates the bias voltage where the device switches from a low conductance to a high conductance branch. The switching bias decreases with the temperature as shown. This is the result of the shifting of Γ -valley electron distribution from the field-emission dominated to thermal-emission dominated regime with the increasing temperature (see Fig. 5.4), reducing the field required to drive the device to switching point.	170

5.6	Schematic diagram of the Si HIWIP device that shows switching behavior. The doping density of the emitter region is $2.5 \times 10^{18} \text{ cm}^{-3}$. Although intentionally undoped, the Si barrier shows some doping migration as photoionization from coulomb wells were observed.	171
5.7	Current voltage curves for the single barrier <i>p</i> -type Si-homojunction device at different temperatures. These were obtained under current bias mode. The sharp jumps in the curves were observed as the field across the $1 \mu\text{m}$ -thick barrier reaches $\sim 15 \text{ kV/cm}$. The switch to the high conductance branch is most likely due to the tunneling of carriers at Fermi level through the barrier. As the temperature was increased, the field required for this reduced slightly as expected, and was described before (see current density spectra in Fig. 5.4).	172
5.8	Current voltage curves for the single barrier <i>p</i> -type Si-homojunction detector with 100 data point average and a data delay of 0.1 s. This was done at different temperatures obtained with the current bias mode. The sudden jump in the curve occurs when the field across the $1 \mu\text{m}$ -thick barrier reaches $\sim 15 \text{ kV/cm}$. The switch to the high conductance ranch is most likely due to the carbon impurities in the barrier. The gap reduces with increasing temperature.	174
A.1	Effect of Savitzky-Golay digital filtering on the responsivity of a detector. The total number of data points used was 21. (a) The contribution to responsivity from top contact without any smoothing and smoothing with (b) a single pass, (c) double passes, and (d) triple passes through the filter.	307
A.2	A combination of number of adjacent points used in the moving window for smoothing, and the number of passes through the filter. N_L and N_R indicate the number of left and right adjacent data points. (a) The contribution to responsivity from top contact without any smoothing and smoothing with (b) a single pass with $N_L = N_R = 20$, (c) a single pass with $N_L = N_R = 30$, and (d) double passes with $N_L = N_R = 20$ through the filter.	308

Chapter 1

Introduction

Interest in Terahertz (THz) research, both in single elements and arrays, led to successful development of many detector technologies. The recent successful development of quantum cascade lasers (QCL) opens the possibility of specialized optical communication in the wavelength range up to $25\text{ }\mu\text{m}$, requiring highly sensitive and fast detectors. Single element high performance far infrared (FIR) ($40\text{--}200\text{ }\mu\text{m}$) semiconductor detectors and large focal plane arrays (FPAs) are used for space astronomy applications, such as NASA's Spitzer space telescope (at launch, known as space infrared telescope facility, SIRTf)¹, and European Space Agency's Herschel Space Observatory (formerly known as FIRST). Far infrared or THz detectors have drawn increased attention for use in ground based, airborne, and space applications. The astronomical interest stems from many different objects which radiate in the FIR: dust disks^{2, 3} and gas molecules such as CO and HD⁴ being important examples. Terahertz detectors will be important in studies of the interstellar medium, planet formation around nearby stars, and of the outer planets and their moons.

There is also potential for use in early universe studies as near- and mid-infrared emissions of high redshift objects can end up in the THz region. In order to better understand the formation of the universe, it is important to study the formation of stars and planets. Although a general picture is accepted, there are many aspects of it that need verification. Whether the outer planets of the Solar System formed by core accretion or disk instability, what initiates the core collapse, and how binary stars are formed are some of the many issues that have remained elusive up to now. For fundamental reasons, FIR and sub-millimeter observations are critical for the study of star and planet formation. To understand the physical processes that occur during star formation, the dominant form of the radiated energy must be observed. Because the stars are born in optically thick interstellar cloud cores, it is very difficult to detect them against the interstellar cloud-background even in the mid-infrared region. Conversion of interstellar clouds into stars occurs in regions of high dust extinction. Far infrared and sub-millimeter wavelengths can penetrate these dusty clouds, facilitating the study of physical processes during star formation. The transitions of the atomic and molecular species that cool the interstellar clouds are in the FIR region. Therefore, study of the spectral signatures of these species is a key to understanding the physical, chemical, and dynamical evolution of the interstellar clouds.

Present and near-future space and air-borne astronomy missions use extrinsic gallium (Ga)-doped germanium (Ge) photoconductor-arrays⁵, and Ge⁶ and silicon (Si)⁷ blocked-impurity-band (BIB) detectors. The mechanical stress required to extend the threshold wavelength of each pixel in an array places limitations in the development of Ge large format arrays. For example, Ge arrays used in recent space missions, such as

Spitzer and Herschel, have a wavelength limit of $210\ \mu\text{m}$. Meanwhile, a GaAs-BIB detector operating up to $300\ \mu\text{m}$ was reported in 2005⁸. The origin of the FIR-response was not from the impurity region, but from the intrinsic blocking layer. Overall, GaAs-BIB detectors are still in the developmental stage. Upcoming astronomy missions, such as SOFIA, SAFIR, Astro-F, and SPICA, demand large detector arrays with longer threshold wavelength.

The non-space applications mostly require detectors operating in the upper THz ($\sim 10\ \text{THz}$ and above) region. In the field of medicine, THz detectors can be used to detect skin cancer, and even to detect asthma like diseases by tracking their bio-markers. Chemical and biological agents can be detected by matching their emitted spectra to fingerprints. These detectors can also be used in aviation to see through fog, in airport security to detect non-metallic explosives, and in many other industrial applications. Technologies developed on other fronts are THz optical mixing in GaAs quantum wells⁹, free-space electro-optic sampling of ZnTe crystals¹⁰, and transition from dissipative to quantum hall states in 2-D electron systems¹¹.

In this dissertation, results from the investigation of THz detectors based on internal photoemission, and the switching mechanism (from low conductance to high conductance branch) of detectors that can lead to pulsing in hot electron diodes are presented. Most of the infrared detectors investigated here are based on GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ heterostructures. In general, the doped dielectric layer (hereafter called “emitter”), used both for infrared photon absorption and hot carrier injection, is degenerate. For photon energies well below the bandgap, the Drude model adequately describes the contribution to dielectric functions from the carrier-plasma. This has been demonstrated through curve-fitting to experimental

reflectance spectra of berillium (Be):doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial films. Although there are two distinct plasmas in p -type emitters and epitaxial films, the entire plasma is treated as consisting of heavy-holes due to the insignificant occupation ($\sim 5\%$) of light-hole states.

The dissertation is organized as follows. **Chapter 2** discusses the infrared detector design, in detail, with the illustration of results supported by models for both homojunction interfacial workfunction internal photoemission (HIWIP) and heterojunction interfacial workfunction internal photoemission (HEIWIP) detectors. The model spectra are generated using an object oriented simulation package developed with C++ (version 6.0, Enterprise edition). The single element detector development was done iteratively to improve the performance and to reduce the threshold frequency down to ~ 1 THz regime. The main considerations for each design stage were (i) enhancing the detector performance through increased responsivity, (ii) increasing efficiency through higher operating temperature, extending the operating range through increased threshold wavelength, etc. For example, the responsivity of the detector can be increased through a resonant cavity architecture (RCA) design. The finer details of this, and other aforementioned factors are discussed along with the experimental results in the relevant chapters. However, it is important to state that the overall goal is to maximize the detectivity of the devices, which have been already tested, and the ones proposed for future. In other words, the goal is to increase the responsivity while minimizing the overall electrical noise of the detector. **Chapter 3** alone covers p -type GaAs homojunction (HIWIP) detectors, and the limitations encountered in extending their threshold wavelength. The HEIWIP detectors are discussed in **Chapter 4**. In **Section 4.4**, infrared absorption of Be:doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ (x is the Al fraction) epitaxial

films in the region 100–1 THz is discussed. Based on the results of this work, an inverted detector structure with a $\text{Al}_x\text{Ga}_{1-x}\text{As}$ emitter and a GaAs barrier was designed and tested. The advantages of this are described in **Section 4.5**.

The switching mechanism in a heterostructure device is discussed in the last chapter. Here, simulation results for switching from a low conductance branch to a high conductance branch in a $\text{GaAs}/\text{Al}_x\text{Ga}_{1-x}\text{As}$ heterostructure are presented.

Chapter 2

Detector Design

2.1 Introduction

The concept of internal photoemission detectors was first proposed for Schottky barrier structures¹². Since then, several other types of detectors^{13, 14, 15} have been demonstrated due to recent developments in molecular beam epitaxy (MBE) and other epitaxial growth technologies. The concept of homojunction internal photoemission FIR detectors was first demonstrated on commercial Si $p-i-n$ diodes¹⁶. Generally, the emitter in a n -type detector is either an impurity band below the conduction band minimum or a semi-metal with its Fermi level above the conduction band minimum. These two types are referred to as Type I and Type II, respectively.

The HIWIP detectors and most of the HEIWIP detectors discussed in this dissertation have doping densities in the range $1 \times 10^{18} - 1 \times 10^{19} \text{ cm}^{-3}$, and therefore belong to the Type II class of internal photoemission detectors. Internal photoemission is the initiating mechanism governing the operation of these detectors. Therefore, various techniques such

as enhancing the optical field using a *mirror-like* dielectric reflector at the bottom of the epitaxial stack (but before the buffer layers, if there are any), locating the emitters within optical field maxima of a desired wavelength, and increasing free carrier absorption through increasing the emitter doping density were tested and are discussed in detail in the following chapters.

An effective way to increase the absorption, hence; the responsivity, is to use RCA in detector designs^{17, 30}. The highly doped bottom contact of the detector will serve as the reflector. Selecting a spectral region marked out for enhancement can be done easily as it only requires fixing the cavity length of the detector. In other words, the cavity spatially distributes the photon absorption probability within the dielectric stack, enabling wavelength-selective optimization of the detector response. However, the limitation arises when enhancement of response at long wavelengths requires a longer resonant cavity, exceeding the thickness limitation for device-quality epitaxial stacks. For example, a 2.5 μm -long cavity in a GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ detector corresponds to a first order resonance at $\lambda = 43 \mu\text{m}$, and using RCA for longer wavelengths requires thicker stacks. For mature materials like GaAs, growing even 10 μm -thick device-quality epitaxial stacks is within the limits of the organometallic chemical vapor deposition (OMCVD) technique.

If RCA is used for wavelength-selective optimization of the detector response, the response across-the-board can be improved using a highly doped buffer layer or a doped substrate. For both the aforementioned cases, a highly doped n -type layer is an automatic choice as it has a lower skin depth than does a p -type layer of similar doping density. If the detector has p -type emitters, the reflector should be electrically isolated in order to avoid a

$p-n$ junction being formed at its interface. On the contrary, the reflector can be used to serve as the bottom contact as well for a detector with n -type emitters.

The background limited infrared photodetector (BLIP) temperature is one of the merit figures of an infrared detector, and is loosely defined as the device temperature where the photocurrent, due to ambient background, matches detector dark current. More precisely, it is the temperature where detectivity is limited by noise arising from the fluctuation of the incident rate of background photons, and not from the inherent noise of the detector. Therefore, BLIP temperature and detectivity, D^* , of a detector depend on dark current mechanisms and responsivity. The responsivity, in turn, depends on carrier relaxation and transport processes. The principles of optimizing the performance of HIWIP and HEIWIP FIR detectors, considering different types of dielectric layers and different wavelength regions, are discussed below.

2.2 Interfacial Workfunction

Generally, the basic structure of a HIWIP (or HEIWIP) detector consists of a top contact (p^{++} or n^{++}) layer, several periods of undoped barrier/ $(p^+$ or $n^+)$ emitter units, and a bottom (p^{++} or n^{++}) contact layer grown on a semi-insulating (SI)-substrate. The detection mechanism mainly involves free carrier absorption in the emitters, followed by the internal photoemission of photoexcited carriers across the interfacial barrier, and then collection of photoemitted carriers at the contact. For simplicity, the structure of a n -type GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ single barrier HEIWIP detector, along with a partial band diagram when reverse biased, is shown in Fig. 2.1. Here, an additional emitter is not necessary as

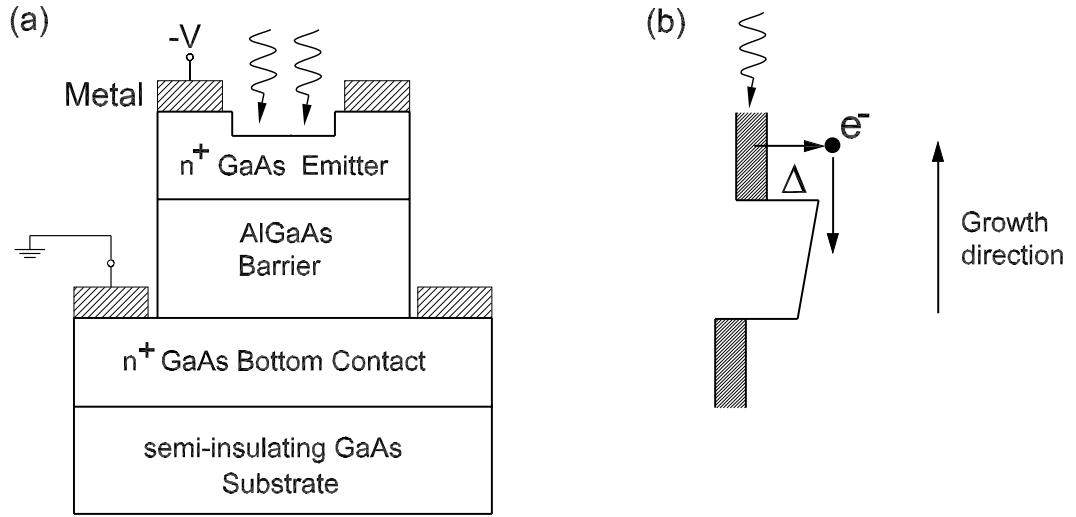


Figure 2.1: (a) Schematic of a single emitter HEIWIP detector after processing. Dielectric layers n^+ -GaAs, i -AlGaAs, and n^+ -GaAs serve as the emitter, barrier and bottom contact, respectively. Here, the emitter is also serving as the top contact layer. (b) Partial energy band diagram showing the interfacial workfunction, and internal photoemission of a carrier. The downward arrow shows the direction of photocarrier drift due to applied bias.

the detector's top and bottom contact will serve as the emitter for reverse and forward bias, respectively. Throughout this dissertation, "detector under forward bias" implies that the top contact is positive relative to the bottom contact, regardless of the dopant type used in the detector. For the n -type detector shown in the figure, the injection of hot carriers is from top contact under reverse bias whereas it is from bottom contact under forward bias. The above two scenarios are exactly opposite for p -type detectors.

The interfacial workfunction of the detector is defined as the minimum energy required for a carrier to undergo internal photoemission. In other words, it is the offset between the Fermi level of the emitter and the corresponding band edge of the barrier (valance band edge for p -type and conduction band edge for n -type). However, the origin of the workfunction differs between homojunction and heterojunction detectors.

2.2.1 HIWIP Detectors

Generally, the interfacial workfunction (Δ) in a HIWIP detector arises due to the difference between band gap narrowing¹⁸ of the highly doped emitter layer and the Fermi level shift of the 3D carrier distribution. When the doping density exceeds the metal-insulator transition (Mott transition), the impurity band is absorbed into the major band (the conduction band for n -type or the valance band for p -type). The shift of the major band is due to the exchange interaction of the majority carriers and impurity-carrier interactions. The Fermion nature of electrons results in the redistribution of carriers to avoid the same spin orientation, decreasing their energy. Band gap narrowing also includes the energy shift due to anisotropy in the conduction band for n -type emitters or the interaction between light and heavy hole bands for p -type emitters. The zero-bias workfunction (Δ) arising from

the major-edge-shift in a HIWIP is shown in Fig. 2.2. This is given by $\Delta = \Delta E_C - E_F$, and the threshold wavelength (λ_0) is given by $\lambda_0 = 1240 \text{ (meV}\mu\text{m)}/\Delta$. However, space charge build up at the interface varies with applied bias, moving the threshold wavelength farther from the design. The sensitivity of such field-variable barrier height to the applied bias depends mainly on the barrier thickness. Although band gap narrowing gives rise to an interfacial barrier in most cases, a Fermi-level shift greater than the narrowing can remove the barrier in others. For example, an emitter doping density of $1 \times 10^{18} \text{ cm}^{-3}$ in a n -GaAs HIWIP, will result in a total negative offset of $\sim 20 \text{ meV}$, removing the interfacial barrier. This is avoidable either by using a differential alloy fraction from emitter to barrier, or by alloying the barrier alone. The latter reduces the detector to a HEIWIP. The variation of Δ with doping density for p -type and n -type GaAs HIWIP detectors is shown in Fig. 2.3(a) and Fig. 2.3(b), respectively. For the p -type GaAs material, the effect on Fermi level shift due to multiple bands has been taken into consideration. The large variation in Fermi level shift in a n -type GaAs is due to lower *density of state mass* of electrons compared to holes. As shown in the figure, doping density greater than $1 \times 10^{17} \text{ cm}^{-3}$ results in a negative offset. A donor doping density above this limit is only possible in n -GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ HEIWIP detectors, where Al composition in the barrier layer provides a leverage in restoring the interfacial workfunction.

2.2.2 HEIWIP Detectors

The workfunction in HEIWIP is given by $\Delta = \Delta_d + \Delta_x$, where Δ_d is the contribution from emitter doping and Δ_x is the contribution from alloy fraction in the barrier. The alloy advantage in heterojunction detectors will allow the tuning of threshold wave-

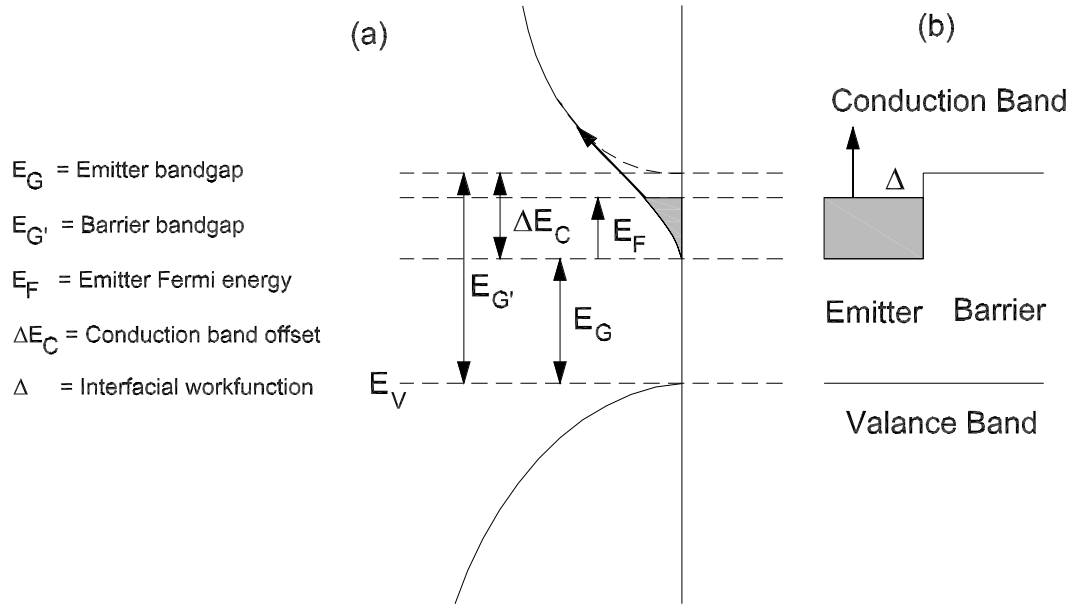


Figure 2.2: (a) Band diagram showing the band gap narrowing and the Fermi level shift in the emitter of a n -type detector. The upper dashed and solid curves correspond to the conduction band of barrier and emitter, respectively. The slanted arrow in the conduction band of the emitter shows an intraband transition. The photoexcitation involves a phonon to conserve momentum. (b) Variation of the conduction band edge and the Fermi level (at $k = 0$) along the growth direction, showing the interfacial workfunction barrier. The vertical arrow corresponds to the photoexcitation shown by the slanted arrow on the left figure.

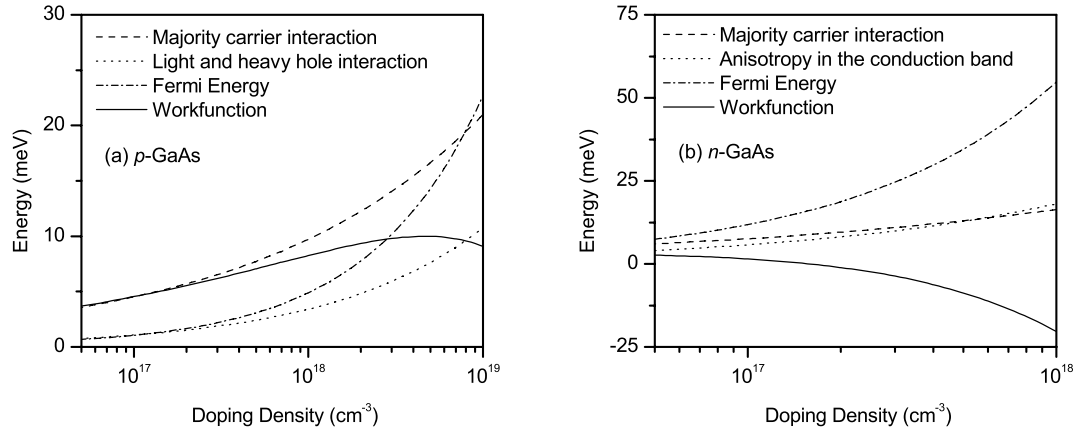


Figure 2.3: (a) Variations of the major band edge due to majority carrier interaction and light-heavy hole interaction in *p*-type GaAs. Also shown is the Fermi level shift including the effects of multiple bands. The change in the workfunction over a doping density change from 1×10^{18} to 1×10^{19} cm⁻³ is $\simeq 1$ meV. (b) Variations of the major band edge due to majority carrier interaction and anisotropy in the conduction band in *n*-type GaAs. The shift in Fermi level in *n*-type GaAs is very large due to low electron *density of state mass*. The workfunction vanishes around a doping density of $\sim 1.5 \times 10^{17}$ cm⁻³ which is the Mott's critical density for a *n*-type GaAs. The contribution to bandgap narrowing from the anisotropy in the conduction band is also shown.

length with a low doping compared to that in a HIWIP. Besides, the HEIWIP has a sharp emitter/barrier interface; hence, the barrier height should not change with applied field.

2.3 Optical Field Distribution in a Dielectric Stack

The complex-amplitude reflection and transmission coefficients for a stack of dielectric layers can be represented by the product of matrices¹⁹ (see “Thin Film Transfer Matrix” in **Appendix A.1**). The optical properties of each layer in the dielectric stack are assumed to be isotropic and homogeneous, and are described by the relative permittivity $\epsilon(\omega)$ and the relative permeability $\mu(\omega)$ of the materials. The matrices describe the transformations of two plane-waves traveling in opposite directions inside the films, and their amplitudes and phase transformations between the films. The schematic diagram of a dielectric stack, with the coordinate system and notations used to obtain the transmission and reflection amplitudes, is shown in Fig. 2.4(a). For clarity, the magnetic field amplitudes at the interface between two dielectric layers (or a dielectric layer and the incident or emerging medium) for oblique-incident *s*-polarized light are shown separately in Fig. 2.4(b).

The electric field distribution in the dielectric layer l can be written as the superposition of two plane waves traveling in opposite directions. For layer l ,

$$\begin{aligned} E_l(x, z, t) &= [E_l^+(z) + E_l^-(z)] \cdot \exp[i(kx \sin \theta - \omega t)] \\ &= [E_l^+ \exp(ik\tilde{n}z) + E_l^- \exp(ik\tilde{n}z)] \cdot \exp[i(kx \sin \theta - \omega t)] \end{aligned} \quad (2.1)$$

where k , ω , and θ are the wavenumber, frequency and the incident angle, respectively, and $\tilde{n} \equiv \tilde{n}(\omega)$ is the generalized complex index of refraction. The generalized refractive index is

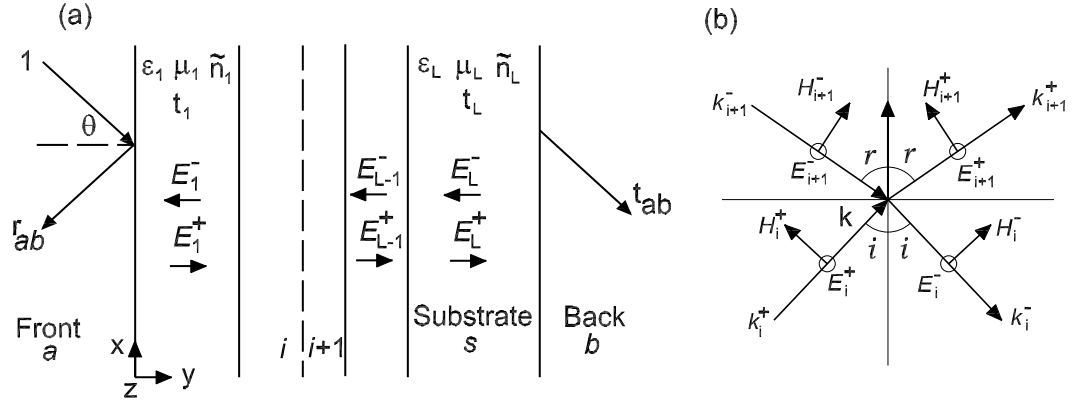


Figure 2.4: (a) A dielectric stack and the electric (E) field components within its layers. Subscripts denote the layer index. For a given dielectric layer, the permittivity, permeability, complex refractive index, and thickness are given by ϵ , μ , \tilde{n} and t , respectively. The incident and the emerging media are labeled as a and b , and θ denotes the incident angle. (b) Magnetic (H) field amplitudes for s-polarized plane waves at the $i/(i+1)$ interface.

given by:

$$\tilde{n} = n + ik = \sqrt{\epsilon\mu - \epsilon_{\text{in}}\mu_{\text{in}} \sin^2 \theta} \quad (2.2)$$

where ϵ and μ are the relative permittivity and relative permeability of the l^{th} dielectric layer, respectively and ϵ_{in} and μ_{in} are that of the incident medium. By applying the matrix method (*i.e.*, by applying the boundary condition at interfaces, and transforming the fields through the layers from right to left), the reflectance and transmittance of the dielectric stack can be determined for the given polarization and incident angle. For a dielectric stack, the transformation of the electric fields across the $l/(l+1)$ interface, and the transformation through layer l are given by:

$$\begin{bmatrix} E_l^+ \\ E_l^- \end{bmatrix}_{z_{l+1}} = \begin{pmatrix} \phi_l^{-1} & 0 \\ 0 & \phi_l \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\tau_{l,l-1}} & \frac{\rho_{l,l-1}}{\tau_{l,l-1}} \\ \frac{\rho_{l,l-1}}{\tau_{l,l-1}} & \frac{1}{\tau_{l,l-1}} \end{pmatrix} \cdot \begin{bmatrix} E_{l-1}^+ \\ E_{l-1}^- \end{bmatrix}_{z_l} \quad (2.3)$$

where $\phi_l = \exp(ik\tilde{n}_l d_l)$, d_l is the thickness of layer l , and ρ and τ are its complex-amplitude reflection and transmission coefficients of the electric field, respectively: they are given by

$$\rho_{l,l-1} = \frac{\tilde{n}_l/\mu_l - \tilde{n}_{l-1}/\mu_{l-1}}{\tilde{n}_l/\mu_l + \tilde{n}_{l-1}/\mu_{l-1}} \quad (2.4)$$

$$\tau_{l,l-1} = \frac{2\tilde{n}_l/\mu_l}{\tilde{n}_l/\mu_l + \tilde{n}_{l-1}/\mu_{l-1}} \quad (2.5)$$

Amplitude variation of the complex reflection and transmission coefficient for a vacuum/ p -GaAs layer interface is shown in Fig. 2.5. The reflection amplitude for both the electric and magnetic components of p -polarized light shows a minimum at, $\theta = 76^\circ$, Brewster's angle for the interface. The corresponding phase variation with incident angle is shown in Fig. 2.6. The dielectric layer used in the calculation has a doping density of $5 \times 10^{18} \text{ cm}^{-3}$, and the wavelength of the incident light is $50 \mu\text{m}$. The above figures together

represent the transformation of electric and magnetic fields of light across the interface. As propagating in a dielectric stack has to emerge from the vacuum side, it is also important to look into the variation of field-coefficients in such a case. The amplitude variation of the complex reflection and transmission coefficients for light penetrating from dielectric layer into vacuum is shown in Fig. 2.7. For the GaAs film layer used in the calculation, both the Brewster's and total internal reflection angle at $\lambda = 50 \mu\text{m}$ are $\sim 11^\circ$. Although the fields exist on the vacuum side for incident angles greater than the total internal reflection angle, the transmittance (or the time-averaged normal component of the Poynting vector) of them will be zero for a layer without any loss mechanism ($k = 0$). In a doped semiconductor layer, the transmittance will be at its minimum, thereby boosting the optical field intensity within the stack.

The complete transformation of the fields through the dielectric stack (from incident side (a) to emerging side (b), shown in Fig. 2.4(a)) yields:

$$\begin{bmatrix} E_i \\ E_r \end{bmatrix}_{z=0} = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} \times \begin{bmatrix} E_t \\ 0 \end{bmatrix}_{z=L} \quad (2.6)$$

$$\begin{bmatrix} 1 \\ r_{ab} \end{bmatrix} = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} \times \begin{bmatrix} t_{ab} \\ 0 \end{bmatrix} \quad (2.7)$$

Thus, the complex-amplitude transmission and reflection coefficient of the dielectric stack can be obtained from the coefficients of the above matrix. The same matrix can be used, without calculating the transfer matrix from bottom to top, to calculate r and t

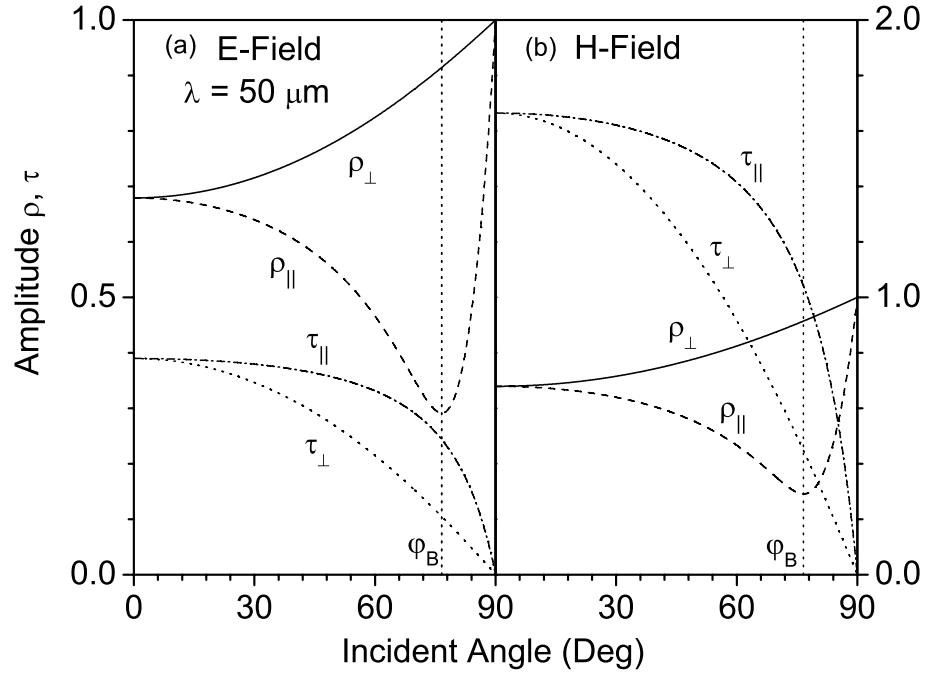


Figure 2.5: Amplitude variation of the reflection (ρ) and transmission (τ) coefficients of (a) electric field and (b) magnetic field with the incident angle. The signs \perp and \parallel refers to s - and p -polarized light, respectively. The light is incident from the vacuum side to a $5 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs epitaxial film; φ_B is the Brewster's angle for the vacuum/film interface.

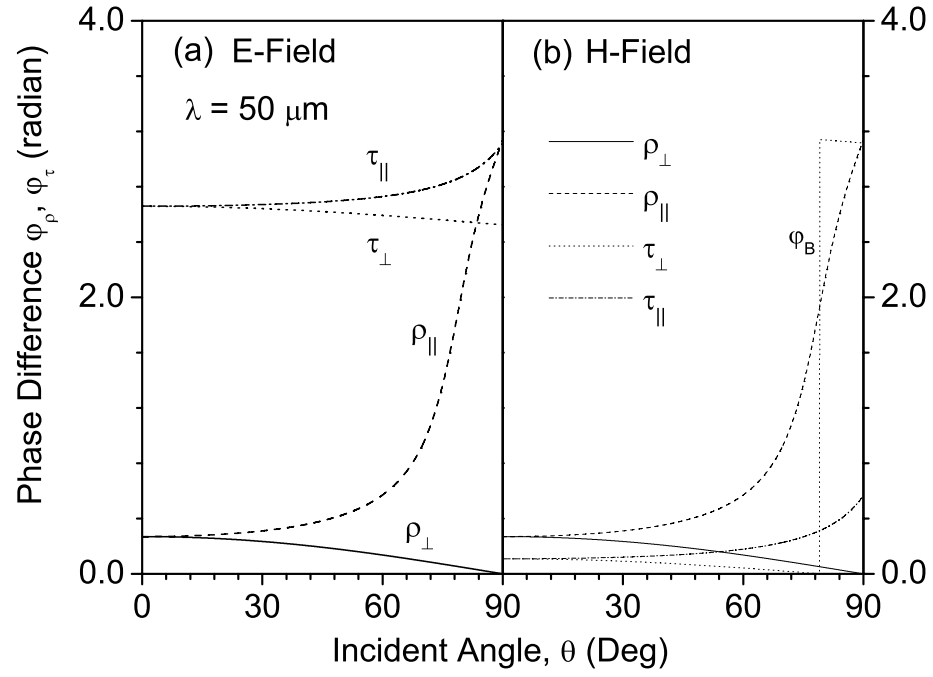


Figure 2.6: Phase variation of the reflection (ρ) and transmission (τ) coefficients of (a) electric field and (b) magnetic field with the incident angle. The light is incident from the vacuum side to a $5 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs epitaxial film: φ_B is the Brewster's angle for the vacuum/film interface; $\varphi_\rho = 0$ implies that the reflected wave is in phase with the incident wave, and $\varphi_\tau = 0$ implies that transmitted wave is in phase with the incident wave.

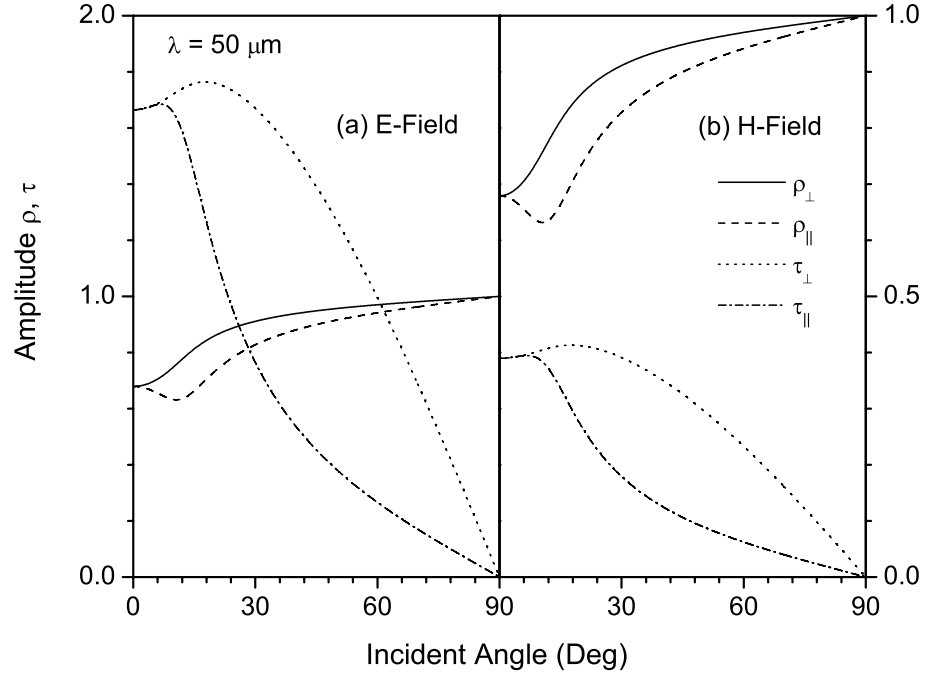


Figure 2.7: Amplitude variation of the reflection (ρ) and transmission (τ) coefficients of (a) electric field and (b) magnetic field with the incident angle. The signs \perp and \parallel refers to s - and p -polarized light, respectively. Radiation penetrates from $5 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs epitaxial film to the vacuum. The Brewster's angle, φ_B , and the angle for total internal reflection for the film/vacuum interface are both $\sim 11^\circ$.

for back-illuminated stack.

$$r_{ab} = T_{21}/T_{11} \quad (2.8)$$

$$t_{ab} = 1/T_{11} \quad (2.9)$$

For back-illumination of the dielectric stack, the coefficients are given by:

$$r_{ba} = -T_{12}/T_{11} \quad (2.10)$$

$$t_{ba} = (T_{11}T_{22} - T_{11}T_{22})/T_{11} \quad (2.11)$$

In order to better understand the energy flow from ambient medium to doped dielectric layers and *vice-versa*, the reflectance and transmittance spectra within the range 5–100 μm are shown in Fig. 2.8. The free carrier absorption and the absorption by lattice vibration have been switched off for clarity. In other words, this demonstrates the reflected and throughput energy of infrared incident on the first interface, as a function of wavelength. While the reflectance of *s*-polarized light increases with incident angle, that of *p* polarized goes through a minimum at the Brewster's angle of the interface. The dip at 272.6 cm^{-1} is due to transverse optical (TO) phonons in the GaAs layer. The variation of reflectance and transmittance with the incident angle, for $\lambda = 50 \mu\text{m}$, is shown in Fig. 2.9. For a GaAs based infrared detector with a top contact doping of $\sim 5 \times 10^{18} \text{ cm}^{-3}$, *p*-polarized light incident close to Brewster's angle will increase the throughput to the detector. Besides, light reaching top contact from below, after undergoing reflection by the bottom contact, will have an incident angle of $\sim 16^\circ$, which is above the total-internal-reflection angle for the GaAs layer/vacuum interface. The small angle will ensure that the incident light beam intercepts the total cross-section area of all the emitters as it propagates through

to the bottom of the stack. The increase in throughput and better reflectance within the structure should improve the optical field strength irrespective of the wavelength. Although this configuration has not been tested in this study, if employed, it would result in better absorption probability. During the optical characterization of the detector, an optical cone is used to increase the intensity to the detector. Coupling the incoming radiation to the cone at $60\text{--}70^\circ$ is intricate and has not been done yet in our cryogenic dewar systems. However, for a detector array, this is not a major issue, as changing the angle can be easily achieved. Unlike in *n*-type quantum well infrared photodetector (QWIP) cameras, where infrared radiation must couple to emitters at $\sim 45^\circ$ from their plane to enhance absorption (no coupling at normal incidence), the interfacial workfunction internal photoemission (IWIP) detector arrays can be operated from normal to very wide incident angles.

The variation of transmittance and reflectance with incident angle for light incident on the ambient medium from the GaAs layer side is shown in Fig. 2.10. The large gap between the spectra of incident angles $\theta = 0$ and 15° is caused when θ exceeds the value for total internal reflection as shown in Fig. 2.11. For a non-absorbing medium (extinction coefficient = 0), the energy flow for incident angles greater than the total-internal-reflection angle would be zero. However, this is not true for a medium with a high damping constant. Figure 2.11 shows the variation of reflectance and transmittance of *s*-polarized (TE-Mode) and *p*-polarized (TM-Mode) radiation. Reflectance for *p*-polarized radiation goes through a minimum whereas the transmittance has a maximum at the Brewster's angle, $\theta = 10.6^\circ$ for the GaAs layer/ambient(vacuum) interface. Here, doping density is so chosen that the total internal reflection occurs at almost the same angle as Brewster's.

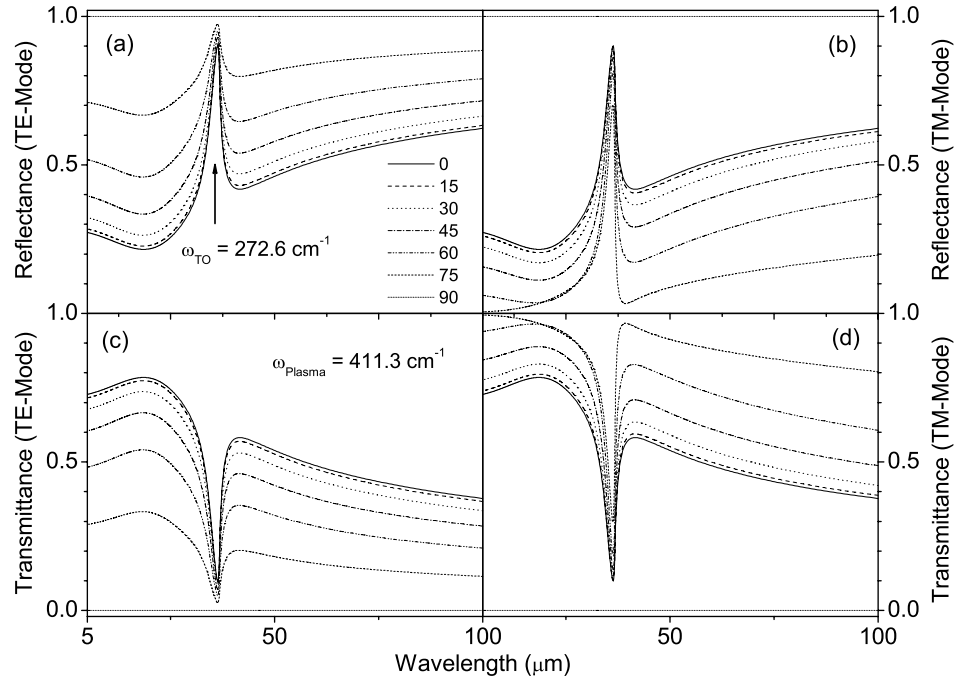


Figure 2.8: Variation of (a), (b) Reflectance and (c), (d) Transmittance spectra with incident angle for *s*-polarized (TE-Mode) and *p*-polarized (TM-Mode) light incident on a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer. The absorption of the dielectric layer has not been considered in this. The dip at 272.6 cm^{-1} is due to transverse optical (TO) phonons in the GaAs layer, and the plasma frequency is 411.3 cm^{-1} .

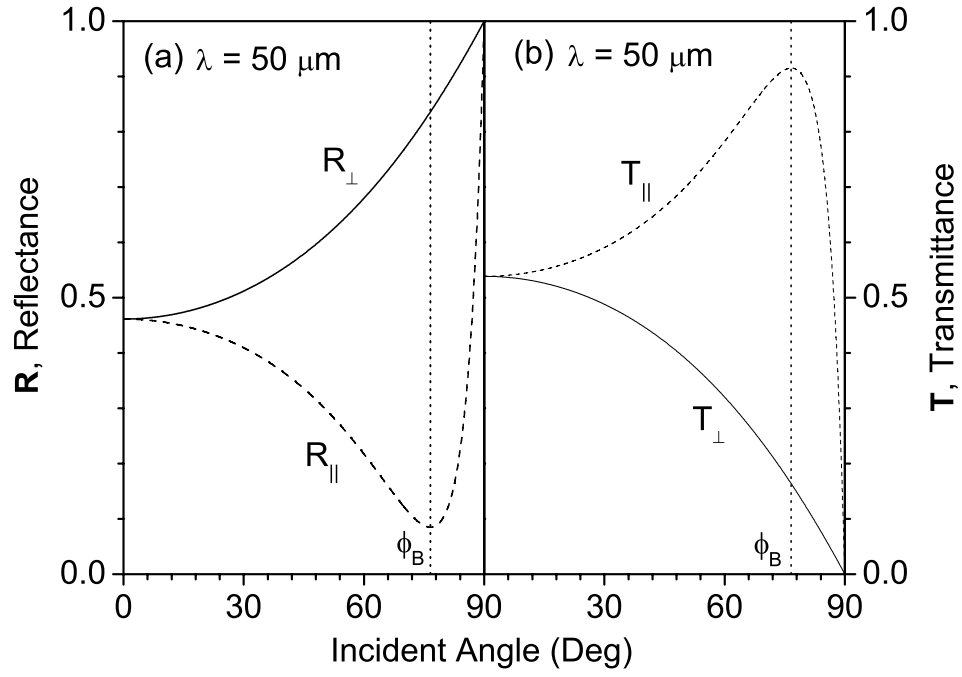


Figure 2.9: (a) Reflectance and (b) Transmittance of *s*-polarized (TE-Mode) and *p*-polarized (TM-Mode) light for $\lambda = 50 \mu\text{m}$. Radiation is incident on a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer from the ambient (vacuum) side. The *p*-polarized reflectance and transmittance reaches the minimum and the maximum, respectively, at the Brewster's angle ($\theta = 76.3^\circ$) for the vacuum/GaAs layer interface.

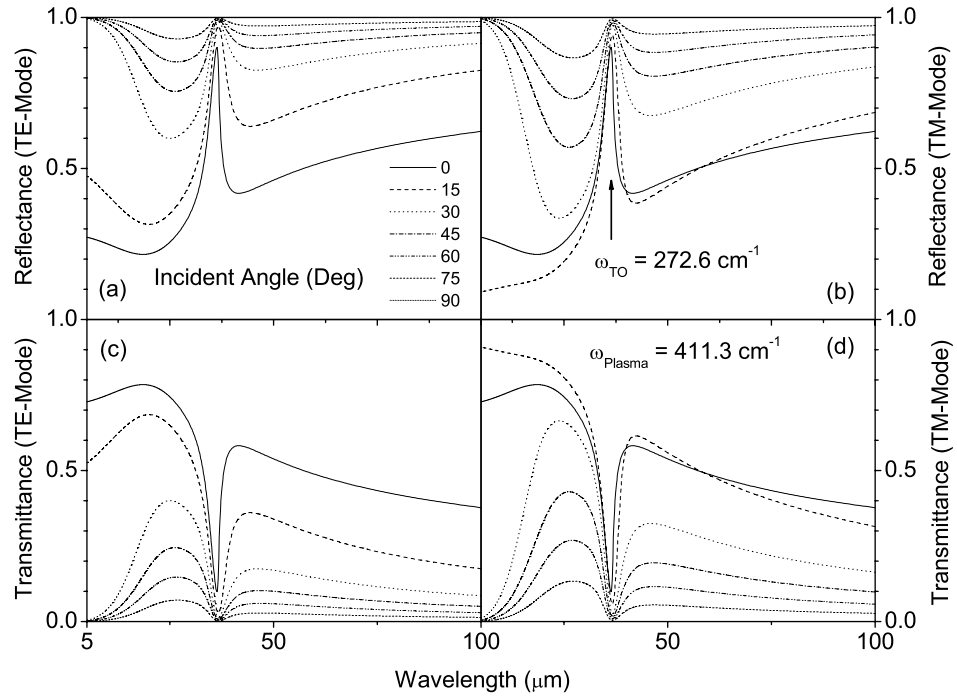


Figure 2.10: Variation of (a), (b) Reflectance and (c), (d) Transmittance spectra with incident angle for *s*-polarized (TE-Mode) and *p*-polarized (TM-Mode) light when infrared light is incident on ambient (here vacuum) from a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer. The dip at 272.6 cm^{-1} is due to transverse optical (TO) phonons in the GaAs layer, and the plasma frequency is 411.3 cm^{-1} . The large gap between incident angles $\theta = 0$ and 15° is due to reflection enhancement from total internal reflection of light.

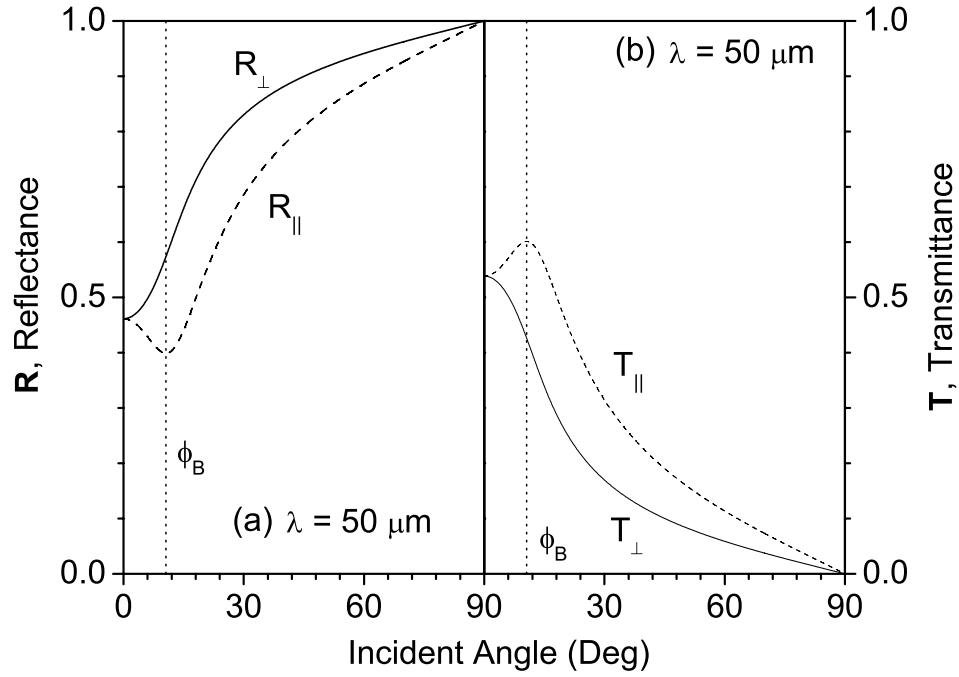


Figure 2.11: Reflectance and transmittance of *s*-polarized (TE-Mode) and *p*-polarized (TM-Mode) light for $\lambda = 50 \mu\text{m}$. Radiation is incident on ambient (vacuum) from a doped ($5 \times 10^{18} \text{ cm}^{-3}$) GaAs dielectric layer side. The *p*-polarized reflectance and transmittance reaches the minimum and the maximum, respectively, at the Brewster's angle ($\theta = 10.6^\circ$) for the GaAs layer/ambient(vacuum) interface. The total-internal-reflection angle is almost the same as the Brewster's angle for the chosen doping density.

Determining the thin film transfer matrix for the given dielectric stack will enable the evaluation of the reflectance and transmittance of the stack.

For s -polarization:

$$\mathbf{R} = r_{ab}^* r_{ab} \quad (2.12)$$

$$\mathbf{T} = (t_{ab}^* t_{ab}) \times \frac{n_{out} \mu_{in}}{n_{in} \mu_{out}} \quad (2.13)$$

and for p -polarization:

$$\mathbf{R} = r_{ab}^* r_{ab} \quad (2.14)$$

$$\mathbf{T} = (t_{ab}^* t_{ab}) \times \frac{n_{out} \epsilon_{in}}{n_{in} \epsilon_{out}} \quad (2.15)$$

Finally, the total absorption by the stack is given by:

$$\mathbf{A} = 1 - \mathbf{T} - \mathbf{R} \quad (2.16)$$

2.3.1 Dispersion Characteristics of Intraband Transitions

As the permeability (μ) of dielectric layers is almost unity, the dispersion of the complex refractive index is mainly determined by their complex permittivity, $n = \sqrt{\epsilon\mu} \simeq \sqrt{\epsilon}$. For materials with a single unit cell, the complex-permittivity in intraband region is described by a single harmonic oscillator²⁰:

$$\epsilon_s = \epsilon_\infty \left[1 + \frac{\omega_{TO}^2 S}{\omega_{TO}^2 - \omega^2 - i\omega\gamma} \right] \quad (2.17)$$

Here, $S = (\epsilon_s - \epsilon_\infty)$ is the oscillator strength, ϵ_s and ϵ_∞ are static and high frequency dielectric constants, respectively, ω_{TO} is the transverse optical (TO) phonon frequency, and γ is the phonon damping constant. The reststrahlen band parameters can be obtained by

non-linear curve fit of reflectance measured in the wavelength range covering the reststrahlen region. If free carriers are present, as in the case of doped GaAs emitters and contacts, the contribution to permittivity from plasma oscillations has to be included, and the complex-permittivity is modified to:

$$\varepsilon(\omega) = \varepsilon_\infty \left[1 - \frac{\omega_p^2}{\omega(\omega + i\omega_0)} \right] + \frac{\omega_{TO}^2 S}{\omega_{TO}^2 - \omega^2 - i\omega\gamma}. \quad (2.18)$$

Here, ω_p is the plasma frequency and $\omega_0 = 1/\tau$ is the free carrier damping constant with τ being the corresponding relaxation time. The frequency of the plasma with effective mass m^* and free carrier density N_p is $\omega_p = \sqrt{N_p q^2 / \varepsilon_o \varepsilon_\infty m^*}$, where q is the magnitude of the electron charge.

The complex-permittivity of an alloy can be described by the additive (Eq: 2.19) or the factorized multi-oscillator model (Eq: 2.20).

$$\varepsilon(\omega) = \varepsilon_\infty \left[1 - \frac{\omega_p^2}{\omega(\omega + i\omega_0)} \right] + \sum_j \frac{S_j \omega_{TO_j}^2}{\omega_{TO_j}^2 - \omega^2 - i\omega\gamma_j} \quad (2.19)$$

$$\varepsilon(\omega) = \varepsilon_\infty \prod_j \frac{\omega_{LO_j}^2 - \omega^2 - i\omega\gamma_j(LO)}{\omega_{TO_j}^2 - \omega^2 - i\omega\gamma_j(TO)} - \frac{\epsilon_\infty \omega_p^2}{\omega(\omega + i\omega_0)} \quad (2.20)$$

Here, j is the oscillator mode, $S_j = \epsilon_\infty(\omega_{LO_j}^2 - \omega_{TO_j}^2)$ and γ_j are the oscillator strength and the damping constant of the j th oscillator, respectively. The number of bands in the dispersion corresponds to the number of different unit cells in the alloy²¹. For example, the dispersion of the dielectric constant for AlGaAs has two bands ($j = 1, 2$): namely, a GaAs-like TO-phonon and an AlAs-like TO-phonon band.

Skin depth: Skin depth ($\delta(\omega)$), at which the optical field decreases to e^{-1} of the field at the interface, is a measure of the penetration ability of light incident on a layer. This depends both on the wavelength (λ) of the incident light and extinction coefficient (k) of the material, $\delta(\lambda) = \lambda/(2\pi k)$. The dispersion of skin depth for n - and p -type GaAs layers with different doping density is shown in Fig. 2.12. For a given doping density, the penetration decreases with the wavelength, reaching almost a constant at long wavelength. As shown, $\delta(\lambda)$ for a n -GaAs layer with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$ is approximately an order of magnitude lower than it is for p -type at long wavelengths. In other words, n -type material has a higher absorption than does p -type. Therefore, for a n -type detector designed to operate at long wavelengths, etching off the top contact will increase the throughput of FIR light to the active region. On the other hand, for a given wavelength and doping density, a n -type layer has a smaller skin depth due to lower effective mass of electrons, resulting in higher reflectivity over p -type layers. Therefore, n -type layers could be used as mirrors inside the structure for wavelength-selective enhancement of photon absorption.

2.3.2 Photocarrier Generation

The responsivity of internal photoemission-type infrared detectors depends on the hot carrier generation in the emitters that inject photocarriers. For example, in a forward-biased p -type HIWIP (or HEIWIP) detector, the injection is from the top contact, and the photocarriers generated in the bottom contact do not contribute to the responsivity. Although the carrier generation in an individual emitter can be readily evaluated through the difference in energy density, $\Delta\{(\vec{\mathbf{E}} \times \vec{\mathbf{H}}) \cdot \hat{n}\}$, for polar materials like GaAs, the absorption by polar phonons has to be offset. This is because, in such cases, the carrier plasma re-

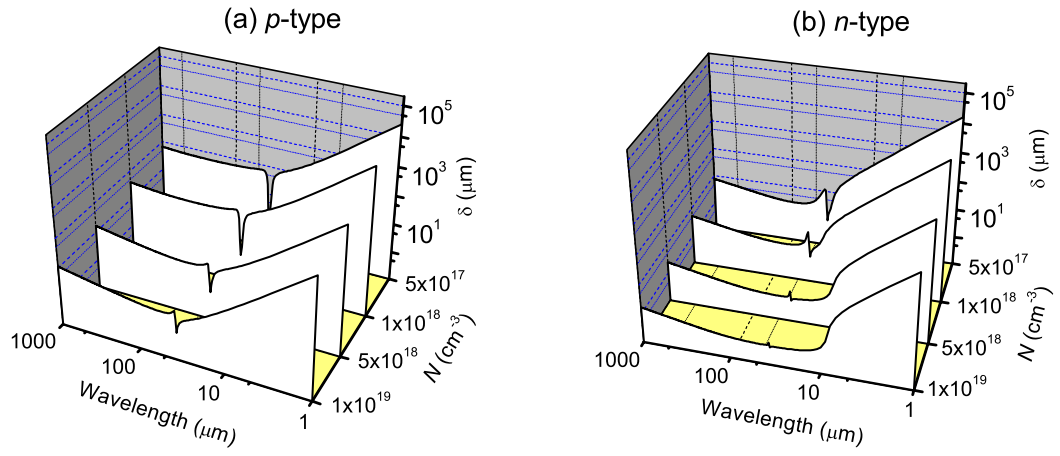


Figure 2.12: Variation of skin depth with doping density for (a) p - and (b) n -type GaAs dielectric layers. Lower skin depth for n -type is due to lower effective mass of electrons compared to holes. The dip around $37 \mu\text{m}$ is due to GaAs TO-phonons.

mains the main contributor to photocarrier generation. The following method is employed to calculate the photocarrier generation from the carrier-plasma alone.

The optical electric fields in the l^{th} layer at a distance z from the incident surface are:

$$\begin{bmatrix} E_l^+(z) \\ E_l^-(z) \end{bmatrix} = \begin{pmatrix} \exp(ik(\omega)\tilde{n}_l z) & 0 \\ 0 & \exp(-ik(\omega)\tilde{n}_l z) \end{pmatrix} \cdot \tilde{Q} \cdot \begin{bmatrix} 1 \\ r \end{bmatrix} \cdot E_0 \quad (2.21)$$

where the matrix \tilde{Q} is the thin film transfer matrix describing the transformation of light up to the l^{th} layer of the epitaxial stack. The hot-carrier generation in the doped layers of the stack is proportional to the optical field distribution given by:

$$|E_j(z)|^2 = (E_j^+(z) + E_j^-(z))^* \cdot (E_j^+(z) + E_j^-(z)). \quad (2.22)$$

The photon absorption probability, η_a , of an emitter (for example j^{th}) is the fraction of the incident photon flux absorbed by the free carriers, and is calculated from the expression:

$$\eta_a(j) = \frac{4\pi}{\lambda} \text{Im}(\varepsilon(\lambda)) \frac{1}{|E_0|^2} \int_0^{W_j} |E_j(z)|^2 dz = \frac{4\pi}{\lambda} \text{Im}(\varepsilon(\lambda)) \frac{\overline{|E_j|^2}}{|E_0|^2} W_j, \quad (2.23)$$

where $\text{Im}(\varepsilon(\lambda))$ is the imaginary part of the dielectric function, λ is the wavelength of the incident radiation, E_j is the optical electric field in the emitter, E_0 is the electric field of the incident radiation, and W_j is the thickness of the emitter layer of interest.

Although the dielectric constant in Eq. 2.17 contains two components, only the first term is included in $\text{Im}(\varepsilon(\lambda))$ in Eq. 2.23, because only the free carriers that are excited by photons will contribute to the photocurrent. The energy absorbed for the generation of optical phonons (second term in Eq. 2.17) dissipates in the crystal without producing hot carriers. However, through its dependence on the optical electric field ($\overline{|E|^2}$), the dispersion

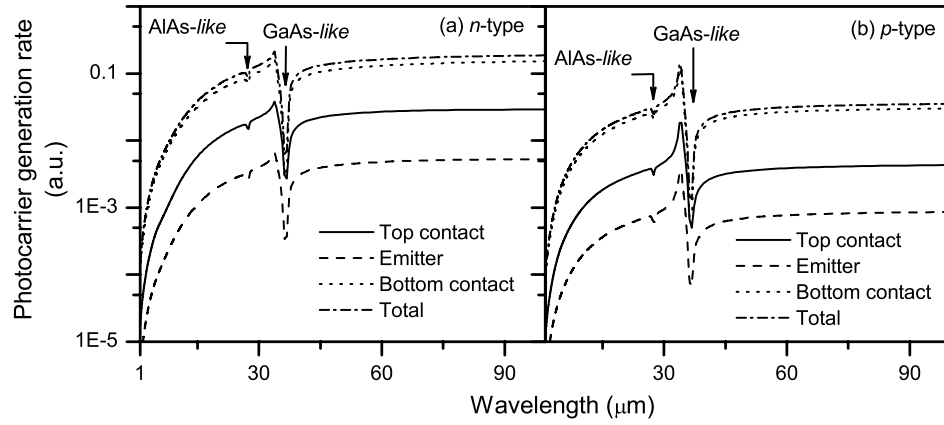


Figure 2.13: Comparison of photocarrier generation rate between the layers in (a) n -type and (b) p -type GaAs/Al_{0.10}Ga_{0.90}As detectors. Each detector has a single emitter with a doping density of $1 \times 10^{18} \text{ cm}^{-3}$. The rate increase in the n -type detector is the result of its better photoabsorption probability due to lower electron effective mass. The generation in the top contributes to photocurrent when the carrier injection is from the top, and that of the bottom contributes when it is from the bottom contact layer.

of η_a still inherits the spectral shape of the total polarization, *i.e.* from both the free carrier-plasma and lattice polarizations.

The comparison of photocarrier generation in the individual layers of a p -type GaAs/Al_{0.10}Ga_{0.90}As and a n -type GaAs/Al_{0.10}Ga_{0.90}As detector is shown in Fig. 2.13. The structures considered have a single emitter, and for clarity, the doping density of both the contacts and emitter is chosen as $1 \times 10^{18} \text{ cm}^{-3}$. Therefore, the variation in carrier generation between the layers of a given structure is mainly due to differences in layer thickness. For both n - and p -type detectors, thickness of the top contact, emitter, and bottom contact are 0.1, 0.02 and 0.7 μm , respectively. Despite similarities between the two types, the n -type has better photocarrier generation rate in its layers. This is the result of a higher absorption coefficient in n -type layers due to lower effective mass of electrons. Therefore, in terms of carrier generation, n -type emitters are more effective than their p counterparts. However, the quantum efficiency of a detector does not depend on carrier generation alone, but it is proportional to the product of both carrier generation and internal photoemission efficiency. The figure shows the carrier generation in top and bottom contacts. The generation from the top and bottom contact layer contributes to photocurrent when the carrier injection is from the top and bottom contact, respectively. For a given structure with the same doping density, for both contacts and the emitter, bottom contact has the maximum generation due its thickness being the largest among all the layers in the detector.

2.4 Internal Photoemission and Hot Carrier Transport

The responsivity of an IWIP detector depends on the internal photoemission quantum efficiency η_i . The ideal emission probability, η_{Ideal} is described by an escape cone model, which is defined as the fraction of excited carriers that has sufficient kinetic energy, associated with the momentum component (p_0) normal to the interface, for emission. In other words, hot carriers directed outside the escape cone, defined by $p_o^2 = 2m^*(E_F + \Delta)$, will emit only if they get redirected into the cone by bulk scattering events or reflection by the emitter film walls. The maximum emission efficiency is defined as the ratio of the number of hot carriers that can be potentially captured to number of photoexcited carriers.

The maximum emission efficiency is defined as the ratio of the number of hot carriers that can be potentially captured to number of photoexcited carriers. For both $\Delta < E_F$ and $\Delta > E_F$ cases, the ideal (η_{Ideal}) and maximum (η_{M}) emission efficiencies of excited carriers are given by the following sets of equations.

For $\Delta > E_F$:

$$\eta_{\text{Ideal}} = 0 \quad (E \leq \Delta) \quad (2.24)$$

$$\eta_{\text{Ideal}} = \frac{3}{4} \cdot \frac{\frac{2}{3} \cdot [(E_F + E)^{1.5} - (E_F + \Delta)^{1.5}] - (E - \Delta) \cdot (E_F + \Delta)^{0.5}}{(E_F + E)^{1.5} - E^{1.5}} \quad (2.25)$$

$$\eta_{\text{M}} = \frac{1}{2} \cdot \frac{(E_F + E)^{1.5} - (E_F + \Delta)^{1.5}}{(E_F + E)^{1.5} - E^{1.5}} \quad (\Delta \leq E < E_F + \Delta)$$

$$\eta_{\text{Ideal}} = \frac{3}{4} \cdot \frac{\frac{2}{3} \cdot [(E_F + E)^{1.5} - (E)^{1.5}] - E_F \cdot (E_F + \Delta)^{0.5}}{(E_F + E)^{1.5} - E^{1.5}} \quad (2.26)$$

$$\eta_{\text{M}} = \frac{1}{2} \quad (E_F + \Delta \leq E)$$

For $\Delta < E_F$, Eq. 2.25 breaks in to two different regions whereas the others remain the same.

$$\begin{aligned}\eta_{\text{Ideal}} &= \frac{3}{4} \cdot \frac{\frac{2}{3} \cdot [(E_F + E)^{1.5} - (E_F + \Delta)^{1.5}] - (E - \Delta) \cdot (E_F + \Delta)^{0.5}}{(E_F + E)^{1.5} - E_F^{1.5}} \\ \eta_{\text{M}} &= \frac{1}{2} \cdot \frac{(E_F + E)^{1.5} - (E_F + \Delta)^{1.5}}{(E_F + E)^{1.5} - E_F^{1.5}} \quad (\Delta \leq E < E_F)\end{aligned}\quad (2.27)$$

$$\begin{aligned}\eta_{\text{Ideal}} &= \frac{3}{4} \cdot \frac{\frac{2}{3} \cdot [(E_F + E)^{1.5} - (E_F + \Delta)^{1.5}] - (E - \Delta) \cdot (E_F + \Delta)^{0.5}}{(E_F + E)^{1.5} - E^{1.5}} \\ \eta_{\text{M}} &= \frac{1}{2} \cdot \frac{(E_F + E)^{1.5} - (E_F + \Delta)^{1.5}}{(E_F + E)^{1.5} - E^{1.5}} \quad (E_F \leq E < E_F + \Delta)\end{aligned}\quad (2.28)$$

Here, E is the energy of the incident radiation, Δ is the interfacial workfunction, and E_F is the Fermi energy of the emitter. The ideal efficiency is modified through scattering processes involving cold electrons (characterized by the scattering length, L_e), inelastic scattering with phonons (characterized by the scattering length, L_p), and multiple reflections of the excited carriers from the surfaces of the emitter (from both the emitter/ambient-medium and emitter/barrier walls). These effects have been incorporated to the emission model^{22, 23}.

The fraction of hot carriers (η_0) captured prior to any bulk scattering events, is well approximated by:

$$\eta_0 = \frac{L^*}{W} \cdot (1 - e^{-\frac{W}{L^*}})^{0.5} \cdot \eta_{\text{Ideal}} \quad (2.29)$$

where $(1 - e^{-\frac{W}{L^*}})^{0.5}$ is the probability of having no scattering from bulk collisions during multiple traversals within the emitter (due to reflections of hot carriers from its walls), and $L^* = \frac{L_e \cdot L_p}{L_e + L_p}$ is the reduced scattering length of a hot carrier. The actual photoemission

probability (η_i) of a hot carrier, taking into account the scattering with cold carriers and phonons, is given by:

$$\eta_i = \eta_0 + \left[1 - \frac{\eta_0}{\eta_M}\right] \gamma \eta_1 + \left[1 - \frac{\eta_0}{\eta_M}\right] \cdot \left[1 - \frac{\eta_1}{\eta_M}\right] \gamma^2 \eta_2 + \dots \quad (2.30)$$

where $\eta_n \equiv \eta_0(E - nh\nu)$ and $\gamma = \frac{L_e}{L_e + L_p}$ is the probability that the hot carrier will collide with a phonon before it collides with a cold electron. Here, n is the number of scattering events after which the thermalization ceases for a given hot carrier.

A comparison of the internal photoemission efficiency of a p -type and a n -type emitter is given in Fig. 2.14. Accordingly, photoemission efficiency is higher for a p -type emitter than it is for a n -type emitter. The difference in η_i is mainly due to the difference in Fermi energy of holes and electrons. For simplicity, the scattering lengths used for the doping density, $\sim 1 \times 10^{18} \text{ cm}^{-3}$, $L_e = 400 \text{ nm}$ and $L_p = 20 \text{ nm}$, are assumed to be independent of both the energy of hot carrier and the lattice equilibrium temperature, although this is not really the case.

2.5 Dark Current and Detector Noise

Dark current in an infrared detector is defined as the current produced when the detector is not exposed to light. In IWIP detectors, it is mainly due to the thermal emission over the interfacial barrier, and the tunneling current through it. The tunneling component of the dark current can be significantly reduced by using barriers with thickness $\sim 0.1 \text{ } \mu\text{m}$ or more, and by operating the detectors at low bias fields ($\leq 1 \text{ kV/cm}$). When increasing the thickness is an option, the thickness should not exceed a maximum for the

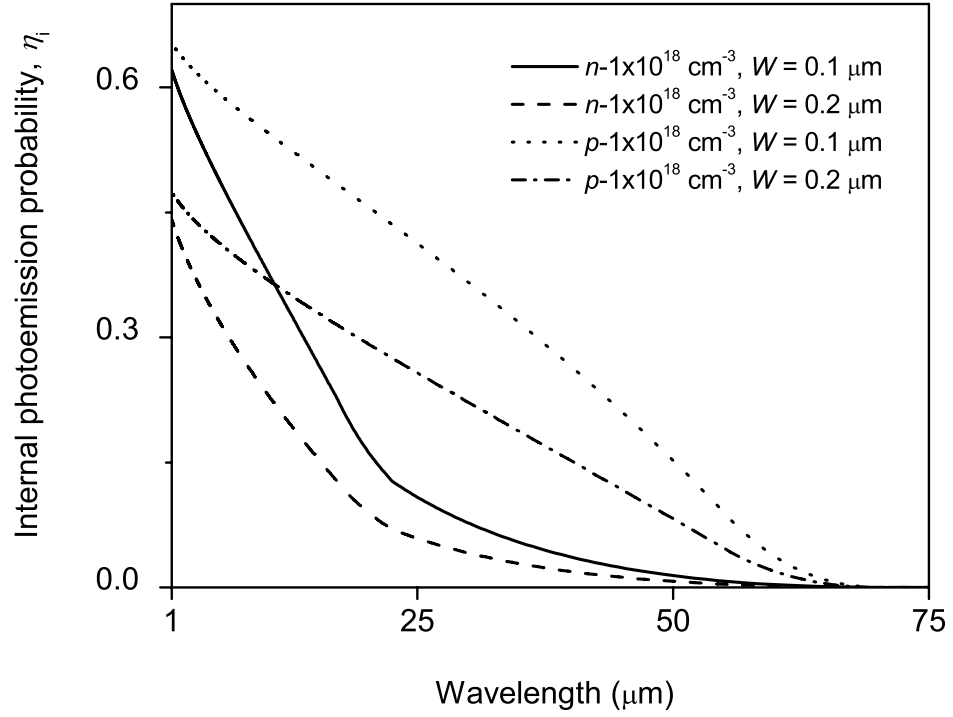


Figure 2.14: Comparison of internal photoemission efficiency, η_i , between n - and p -type GaAs emitters with a doping density of $1 \times 10^{18} \text{ cm}^{-3}$. Interfacial workfunction of the barrier is set to $\Delta = 17.7 \text{ meV}$ to obtain a threshold of $\lambda_0 = 60 \mu\text{m}$. Photoemission efficiency is better in p -type GaAs emitters due to higher effective mass of its carriers. The efficiency decreases with the increasing emitter thickness, and is also a function of the scattering lengths. Generally, the energy independent values of $L_e = 400 \text{ nm}$ and $L_p = 20 \text{ nm}$ are used for doped GaAs films.

given barrier material, because the effort may be counterproductive due to possible defect formation in such thick barriers. A detailed description of the tunneling mechanism is given in **Chapter 5**. The thermionic current in a IWIP detector is well described in the frame of the 3D-drift model²⁴ given by:

$$I_{dark} = qA \frac{\mu F}{[1 + (\mu F/v_{sat})^2]^{1/2}} 2(m^* kT/2\pi\hbar^2)^{3/2} \exp(-(\Delta - \alpha F)/kT), \quad (2.31)$$

where μ is the carrier mobility, F is the applied field, $v_{sat} \sim 10^7$ cm/s is the terminal drift-velocity of the carriers, Δ is the interfacial workfunction, and α (~ 100 – 200 Å) is a barrier lowering parameter that is sensitive to the applied field.

To demonstrate the compatibility of the model, the experimental and calculated dark currents at different temperatures, for HE0204 (see Table 2.1 for detector parameters), are shown in Fig. 2.15. The interfacial workfunction, $\Delta = 77$ meV at low bias, was determined from the Arrhenius plot for the detector dark current. This agrees well with the spectral threshold of $16 \mu\text{m}$ ²⁵. For all the temperatures shown, a barrier lowering parameter of $\alpha \simeq 180$ Å was used to obtain the model fit to the experimental dark current. The figure shows that the dark current can be well described by the above model, and the deviation from the model at higher bias is attributed to the increase in tunneling probability. For high temperatures and low field strengths, emission over the barrier predominates whereas for high field strengths and low temperatures, tunneling through the barrier predominates in generating the dark current.

Noise current: The noise current of the detector is related to the mean current (\bar{I}) through it by²⁶:

$$I_{noise}^2 = 4q\bar{I}g_nB, \quad (2.32)$$

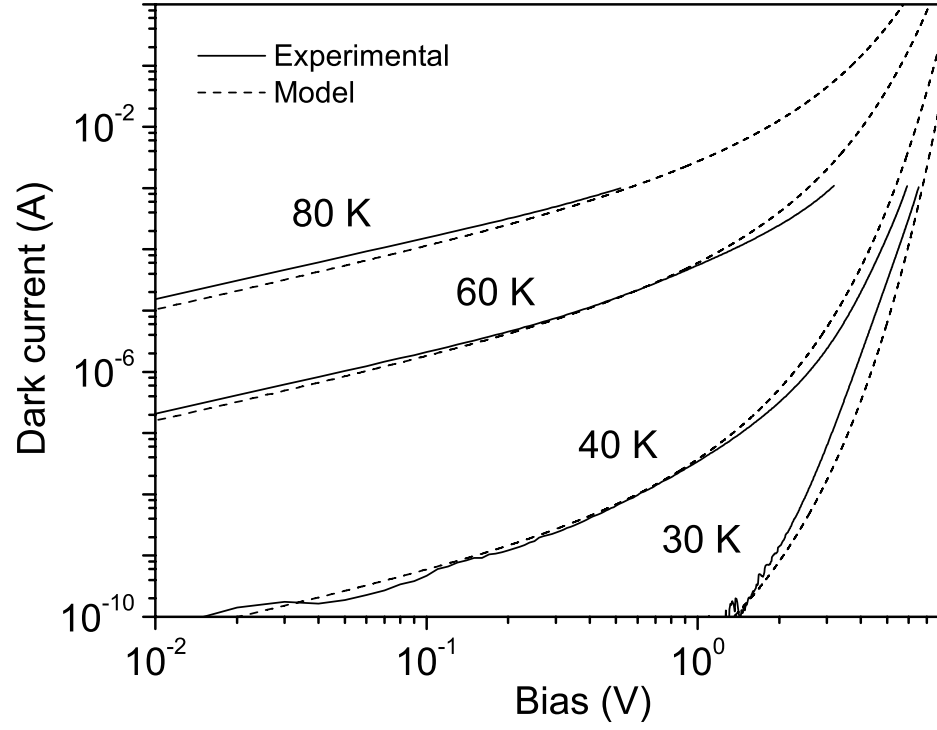


Figure 2.15: Experimental and model bias dependence of the dark current for detector HE0204 at temperatures where the thermionic component becomes dominant. The interfacial workfunction, $\Delta=77$ meV, used in the model was derived from the Arrhenius plot for the detector. The deviation at high bias is due to the increase in the tunneling probability.

where g_n is the noise gain and B is the measurement bandwidth. According to²⁷, the noise gain g_n and photocurrent gain g_p are:

$$g_n = \frac{1 - p_c/2}{p_c(N + 1)} + \frac{1 - (1 - p_c)^{N+1}}{p_c(N + 1)^2[1 - (1 - p_c)^N]} \quad (2.33)$$

$$g_p = \frac{1}{p_c N} \quad (2.34)$$

where p_c is the capture probability of a carrier traversing a given emitter, and N is the number of such emitters in the detector. The ratio g_n/g_p varies from 0.5 to 1 as p_c varies from 0 to 1 and N from 1 to ∞ . Therefore, one may ignore the difference between them in many applications. If the capture probability is low ($p_c \ll 1$), the difference between the noise and photocurrent gains is negligible, and they are given by the expression $g = g_n = g_p = 1/p_c N$. The ratio approaches $1 - p_c/2$ for $N \gg 1$.

The capture probability is defined as the ratio $p_c = (1/\tau_{rec})/(1/\tau_{rec} + 1/\tau_s) = \tau_s/(\tau_s + \tau_{rec})$, where τ_s is the carrier sweep-out time and τ_{rec} is the recombination time (lifetime). For $\tau_s \ll \tau_{rec}$, the capture probability p_c is low, and the gain reduces to the conventional photoconductivity expression of:

$$g = \tau_{rec}/N\tau_s = \tau_{rec}/\tau_{tr} = \mu\tau_{rec}F/d, \quad (2.35)$$

where τ_{tr} is the transit time, μ is the carrier mobility, F is the applied bias field, and d is the effective device thickness of the structure.

2.6 Responsivity

The spectral responsivity of an infrared detector is a measure of its response to radiation at a specified wavelength. For IWIPs, the interfacial workfunction is a prerequisite for forecasting their response spectra, for this allows the sequential evaluations of optical field distribution in the dielectric stack, hot carrier generation in emitters and contacts, ideal photoemission efficiency, thermalization of hot carriers due to phonons, and loss during (hot carrier) transport. In brief, the total quantum efficiency²⁸ is the product of photon absorption η_a , internal photoemission η_i , and hot-carrier collection η_t probabilities, so $\eta = \eta_a \eta_i \eta_t$. The responsivity of the detector is then given by $R = q\eta\lambda/hc$.

Designing a detector with the best performance for a particular application is done through a simulation process. Here, several detector structures were considered to demonstrate the effects of changing the parameters on their performance. The structure parameters of the detectors, considered for this purpose, are shown in Table 2.1.

Structure I consists of n^{++} top and bottom contacts with an undoped barrier grown on a SI-GaAs substrate. The total absorption spectra of this structure as a function of bottom-contact thickness are shown in Fig. 2.16(a). The two absorption peaks in the spectra correspond to the first and the third order cavity resonance for wavelengths $\lambda = 15$ and $7 \mu\text{m}$, respectively. The ripples in the spectra are due to Fabry-Pérot interference in the bottom contact layer. The intensity around the third order peak shows a strong dependence on the bottom contact thickness. This is due to the rapid ($\sim 1000 \mu\text{m}$) change in the skin depth of the bottom contact layer (which has a donor doping density of $1 \times 10^{19} \text{ cm}^{-3}$) over the wavelength range $1\text{--}10 \mu\text{m}$ (see Fig. 2.12). As the bottom contact

Table 2.1: The structure parameters of detectors considered to demonstrate the effects of vital parameters on the detector response. The last two listed have already been tested, and are used here for the discussion's continuity. The top and bottom contacts of the structures are highly doped (denoted by the superscripts $++$) GaAs, and the barriers are undoped $\text{Al}_x\text{Ga}_{1-x}\text{As}$. The substrate is either highly doped or SI-GaAs. Notations W , N , and λ_0 denote the layer thickness, emitter/barrier units and the threshold wavelength of the detectors, respectively.

Design or Sample	Top contact		Emitter layer		Barrier layer	Bottom contact		Substrate type	N	λ_0 (μm)
	Type	W (μm)	Type	W (nm)	W (nm)	Type	W (μm)			
I	n^{++}	0.2	—	—	200 1000	n^{++}	0.4–10 10	SI	—	25
II	n^{++}	0.2	—	—	1000	—	—	n^{++} n^{++}	— 1	25
III	p^{++}	0.4	p^+	70	100	p^{++}	0.7	n^{++}	9	15
IV	p^{++}	0.1	p^+	30	50	n^{++}	0.7	SI	22	35
HE0204	p^{++}	0.1	p^+	19	125	p^{++}	0.7	SI	16	16
2409	p^{++}	0.2	p^+	15	77	p^{++}	0.73	SI	30	70

thickness is increased the attenuation of the short wavelength radiation increases, resulting in increased photocarrier generation. This is illustrated in Fig. 2.16(b). The figure shows the detector responsivity in the forward and reverse configurations for the chosen bottom contact thickness of $W_b = 0.7 \mu\text{m}$. The cavity resonance causes reverse bias response to be larger than for the forward bias. The inset to Fig. 2.16(b) shows that increasing the bottom contact thickness from $W_b = 0.7$ to $10 \mu\text{m}$ (a factor of ~ 140) only increases the reverse bias response around $15 \mu\text{m}$ by a factor of ~ 1.5 and the short wavelength region is not increased at all. Whereas for forward bias, the increase in the absorption around the short wavelength region does not result in improving the photocurrent. Although the absorption probability is high, prior to photoemission, a sizeable fraction of the hot carriers is attenuated by scattering off the bottom contact walls before photoemission. Besides, if the thickness is very large, as in this case, then the number of scattering events (both $e-e$ and $e-p$) for a given hot carrier increases, and its energy falls below the required minimum for photoemission upon arrival at the interface. Therefore, it is preferred that the emitter thickness should be at least on the order of the scattering length ($L_p = 20 \text{ nm}$ for GaAs), for a given material. Although the emitters in the detectors tested so far are below this thickness, wet etching employed in device fabrication requires the bottom contact thickness to be $\sim 30 \%$ of the stack thickness to ensure electrical contact. For example, a $2.5 \mu\text{m}$ -thick GaAs detector structure needs $\sim 0.7 \mu\text{m}$ -thick bottom contact layers.

The cavity length of a detector can be changed by varying the active thickness of the device excluding the bottom contact. This will result in the redistribution of the resonance peaks as shown in Fig. 2.16(c). As shown, the main peak can be tuned from $\lambda=15$

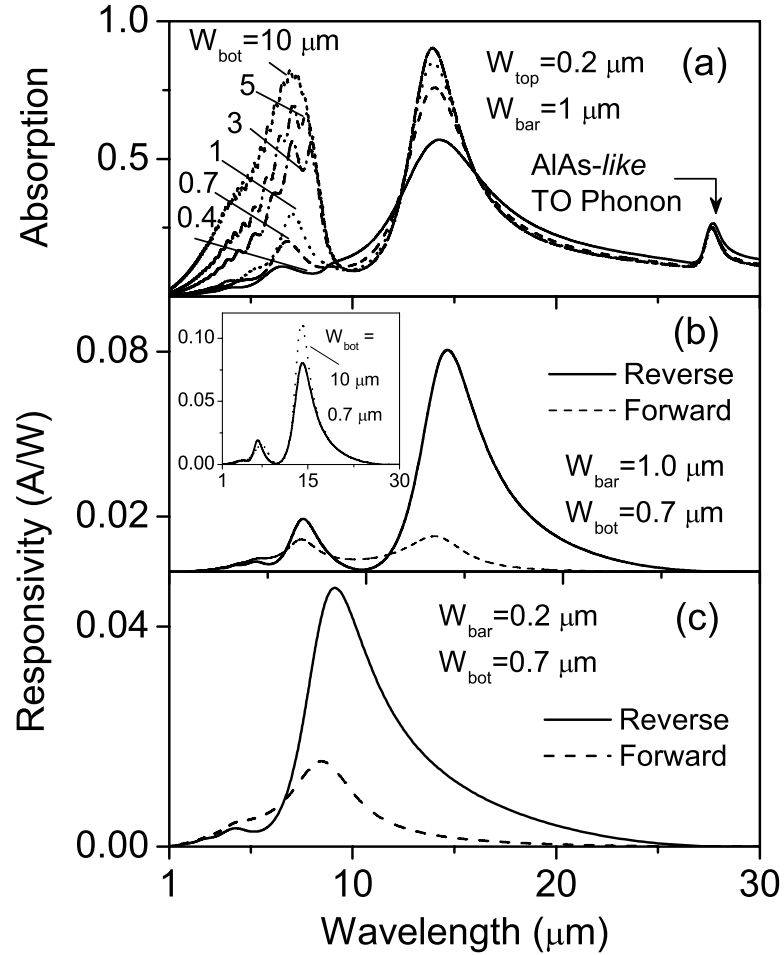


Figure 2.16: (a) Photon absorption as a function of bottom contact thickness for a single barrier n -type GaAs/Al_{0.26}Ga_{0.74}As detector (structure I). Doping density of the contacts are $1 \times 10^{19} \text{ cm}^{-3}$. The absorption around the short wavelength region increases as the thickness (W_{bot}) of the bottom contact is increased. The ripples in the spectra are due to Fabry-Pérot interference in the bottom contact layer. (b) Detector response for $W_{\text{bot}} = 0.7 \mu\text{m}$. The left and right response peaks correspond to the third and the first order resonance, respectively. The threshold wavelength is $\lambda_0 = 27 \mu\text{m}$. The inset shows the insignificant increase in the reverse bias response over a large increase in the bottom contact thickness: from $W_b = 0.7$ to $10 \mu\text{m}$ (a factor of ~ 140). (c) Detector response for $W_{\text{bar}} = 0.2 \mu\text{m}$. The reduced epitaxial stack thickness decreases the cavity length, which results in the redistribution of the resonance peaks.

to $8\ \mu\text{m}$ by varying the barrier thickness from $W_b=1.0$ to $0.2\ \mu\text{m}$. Therefore, detectors with different peak positions can be designed by maintaining all the other parameters, provided that the barriers are still sufficiently thick to minimize the tunneling current of the detector. The Al fraction on the structure was selected such that the threshold wavelength will be at $27\ \mu\text{m}$, which is just below the AlAs-like TO-phonon absorption.

A broad band detector can be designed using a n -type substrate. The hot carrier generation in structure II with a n -type top contact (of doping density $2\times 10^{18}\ \text{cm}^{-3}$) and an undoped barrier layer grown on a highly doped n -type substrate (also of doping density $2\times 10^{18}\ \text{cm}^{-3}$) is shown in Fig. 2.17(a). Here, the doped substrate can be used as the bottom contact of the detector as well. The hot carrier generation is similar in shape to the absorption probability spectra and differ only by a scaling factor. The detector shows a high absorption in a broad spectral range, from 5 to $25\ \mu\text{m}$. The dip at $\sim 28\ \mu\text{m}$ is due to the AlAs-like TO-phonons in the $\text{Al}_{0.11}\text{Ga}_{0.89}\text{As}$ barrier. Unlike in the previous design, where the emitter doping density is $1\times 10^{19}\ \text{cm}^{-3}$, here the carrier generation is improved throughout the range due to increased reflection from the doped substrate. For the same reason, the difference between the peak generation corresponding to cavity resonance and the minimum corresponding to the destructive condition is less than in the previous design. The total carrier generation as a function of top contact thickness is shown in Fig. 2.17(b). The change in the peak positions is due to the changing cavity length of the detector. As the top contact thickness is increased the cavity positions should shift towards longer wavelengths. The shift pattern of the peaks located around the two phonons does in fact shows a negative trend as they merely represent the residual of the first order cavity

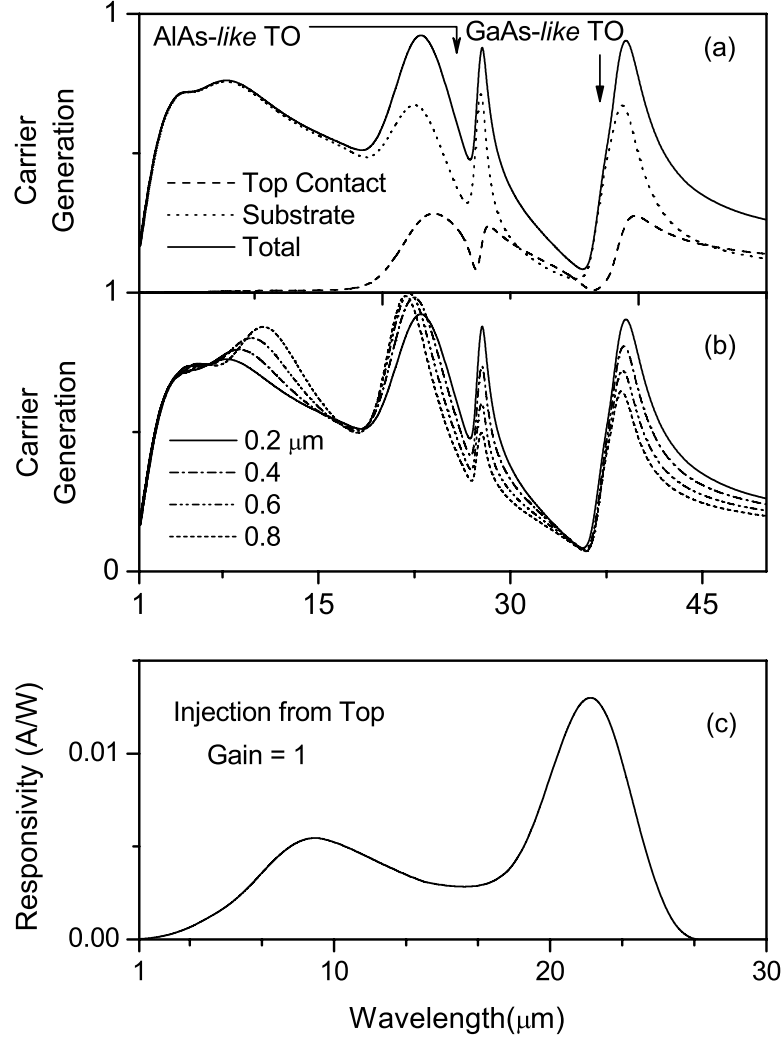


Figure 2.17: (a) Photocarrier generation rate for a single barrier detector (structure II) with a n -type top contact and an undoped barrier grown on a n -type substrate. The doping densities of the top contact and substrate are both $2 \times 10^{18} \text{ cm}^{-3}$, and the $\text{Al}_{0.11}\text{Ga}_{0.89}\text{As}$ barrier thickness is $1 \text{ } \mu\text{m}$. Here, the substrate serves as the bottom contact. The arrows indicate the GaAs-like and AlAs-like TO-phonon effects. The generation of photocarriers in the substrate is within three skin depth scales for a given wavelength. (b) Total carrier generation as a function of top contact thickness. The carrier generation around the designed peak slowly increases with the contact thickness. (c) Responsivity spectra for $W_{\text{top}} = 0.2 \text{ } \mu\text{m}$ under reverse bias. The carrier injection is from the top contact. The interfacial work function corresponds to $\lambda_0 \sim 27 \text{ } \mu\text{m}$.

masked by the phonon absorption. The responsivity of the designed detector is shown in Fig. 2.17(c). As the increase in carrier generation is low compared to the increase in the top contact thickness, the top contact thickness is fixed at $0.2 \mu\text{m}$. The main absorption in the structure occurs in the substrate which leads to low response for a forward bias. This occurs due to the very large skin depth corresponding to the free carrier density of $\sim 10^{18} \text{ cm}^{-3}$.

The variation in photon absorption probability among the individual emitters of structure III is shown in Fig. 2.18. As shown, the first emitter has a maximum absorption probability around $10 \mu\text{m}$, and the 9th has its maximum around $15 \mu\text{m}$. The *n*-type substrate enhances the reflection of radiation of $\lambda > 10 \mu\text{m}$ while allowing radiation of shorter wavelengths to be transmitted. This attenuates radiation of $\lambda < 10 \mu\text{m}$, producing the main absorption within the substrate. The thin emitters mean that the absorption in a given emitter is less than the absorption in the top or bottom contact. On the other hand, the scattering length of the hot carriers in the emitter is on the order of the emitter thickness, leading to a high internal photoemission efficiency. Therefore, the contribution to the photocurrent from the emitters remain high. Although individual emitters selectively maximize the absorption probability at different wavelengths, the detector response will not show a prominent resonance signature due to the collective contribution to photocurrent from all the emitters. The responsivity spectra of the detector illustrating this fact are shown in Fig. 2.19. Although the absorption of the top or bottom contact was not shown in Fig. 2.18, these response spectra also include the carrier generation arising from absorption in the top or bottom contact. Generally, a detector designed for $\lambda_0 = 15 \mu\text{m}$ or less allows the detector to operate at liquid Nitrogen temperature.

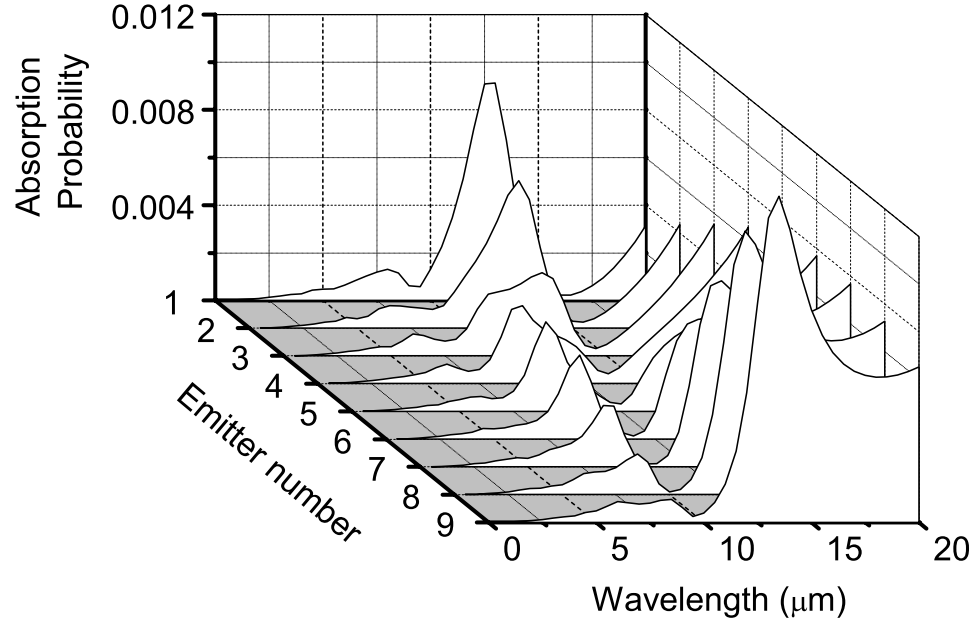


Figure 2.18: Calculated absorption probability in the emitters of structure III. The structure consists of a 400 nm-thick *p*-type top contact and 700 nm-thick *p*-type bottom contact of doping density $1 \times 10^{19} \text{ cm}^{-3}$. This has nine emitter/barrier units. The 70 nm-thick emitters are *p*-type with a doping density of $3 \times 10^{18} \text{ cm}^{-3}$, and the 100 nm-thick barriers are undoped. The substrate is *n*-type with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$.

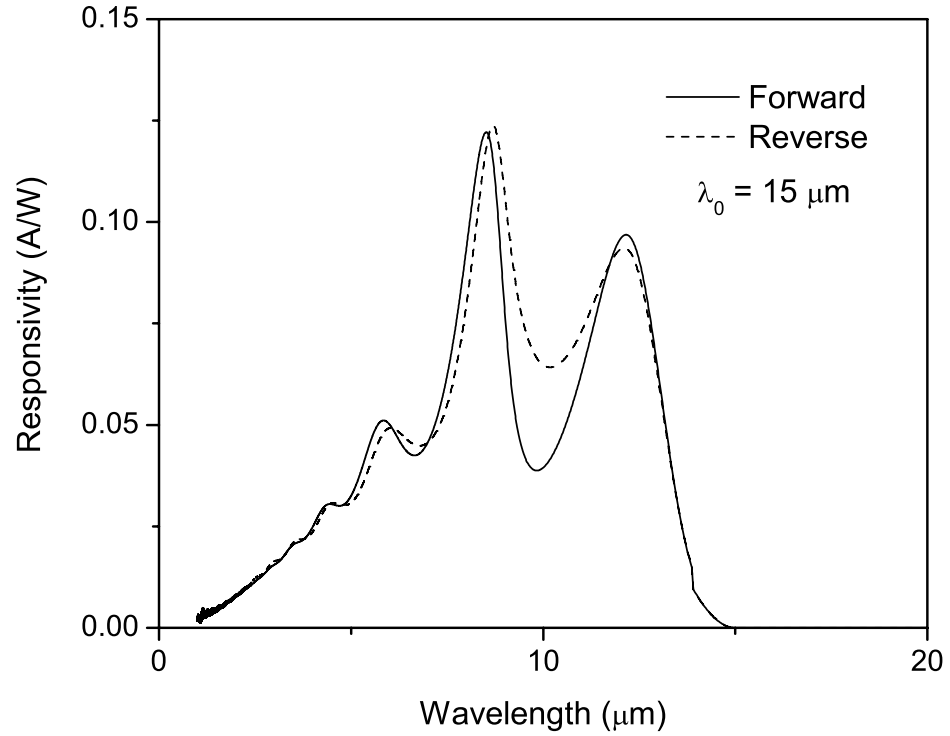


Figure 2.19: Calculated responsivity spectra for structure III with nine emitter/barrier units. The structure consists of *p*-type top and bottom contacts with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$, and a thickness of 400 and 700 nm, respectively. The 70 nm-thick emitters are *p*-type with a doping density of $3 \times 10^{18} \text{ cm}^{-3}$, and the 100 nm-thick barriers are undoped. The *n*-type substrate has a doping density of $1 \times 10^{19} \text{ cm}^{-3}$. The forward and reverse spectra represent the carriers photoinjected from the top and bottom contacts, respectively, with all the emitters contributing to both bias directions.

The thermionic dark current for structure III, designed for $\lambda_0 = 15 \mu\text{m}$ ($\Delta = 83 \text{ meV}$), is shown in Fig. 2.20. The detector bias is 2 kV/cm and the hole mobility used in the 3D-drift model for dark current is $\mu = 60 \text{ cm}^2/\text{Vs}$. Also shown is the photocurrent at 300 K background, assuming $\eta = 1$ and $g_n = g_p = 1$, for field of views (FOVs) of 60° and 180° . The maximum obtainable BLIP temperature under the respective FOVs are $T_{\text{BLIP}} = 75$ and 82 K . The corresponding specific detectivity limits are 6.6×10^{10} and $3.3 \times 10^{10} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, respectively. In reality, η depends on the wavelength and this leads to a BLIP temperature below the ideal maximum. The detectors (with a similar threshold) characterized so far give a BLIP temperature $\sim 45 \text{ K}$.

The main absorption in structure III occurs in the bottom contact whereas the emitters absorb only a small part of the radiation. Significant improvement may be achieved in the structure with p -type top and n -type bottom contacts. The structure IV is designed for a peak response around $22 \mu\text{m}$ (which can be used to detect radiation from a quantum cascade laser²⁹). A 35 meV ($\lambda_0 \simeq 35 \mu\text{m}$) interfacial work function was chosen to prevent high reflection in the reststrahlen band. The n -type bottom contact serves both as an electric contact and a reflector. Figure 2.21(a) demonstrates that the absorption in the emitters increases with wavelength, reaching a value $\sim 70\%$ around $24 \mu\text{m}$. This manifests as a response peak as shown in Fig. 2.21(b). The change in the peak position is due to the decrease in photoemission with wavelength. The peak-response wavelength can be tuned by changing the stack thickness as shown in Fig. 2.21(b). Increasing the number of emitter/barrier units would increase the total thickness of the structure, shifting the resonance peaks towards longer wavelengths. As a side effect, the amplitude of the peak response

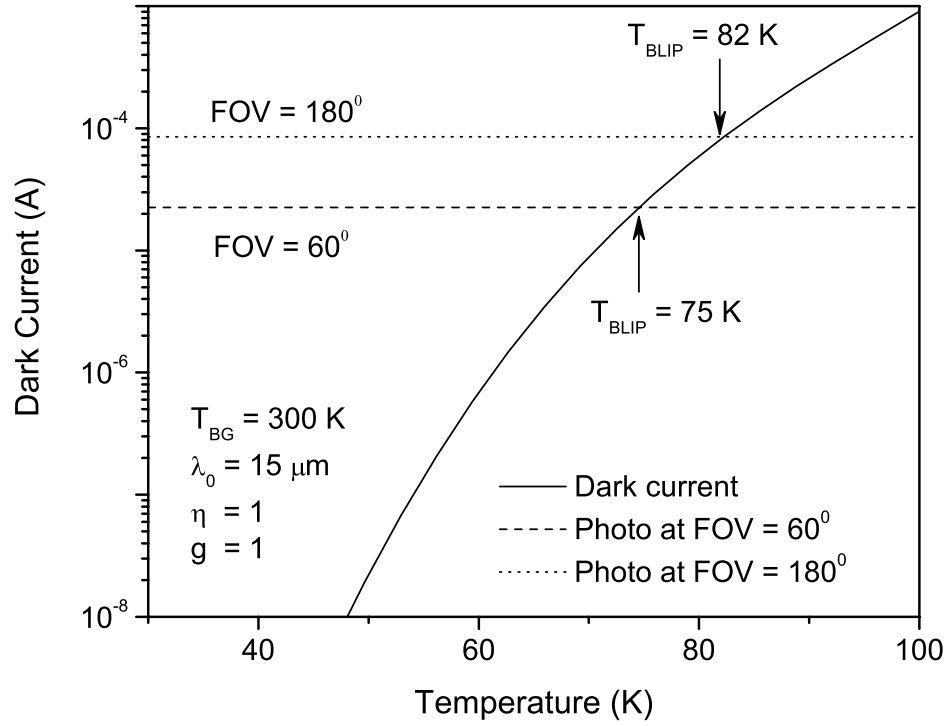


Figure 2.20: Calculated temperature dependence of dark current in structure III. The applied bias is 2 kV/cm, and the interfacial workfunction corresponds to $\lambda_0 = 15 \text{ } \mu\text{m}$. Photocurrent levels of 8.5 and 21 μA correspond to 300 K background illumination at FOV = 60° and 180°, respectively. The quantum efficiency (η) and gain are assumed to be unity. The Intersections of the curves show that $T_{BLIP} = 75$ and 82 K for the respective FOVs.

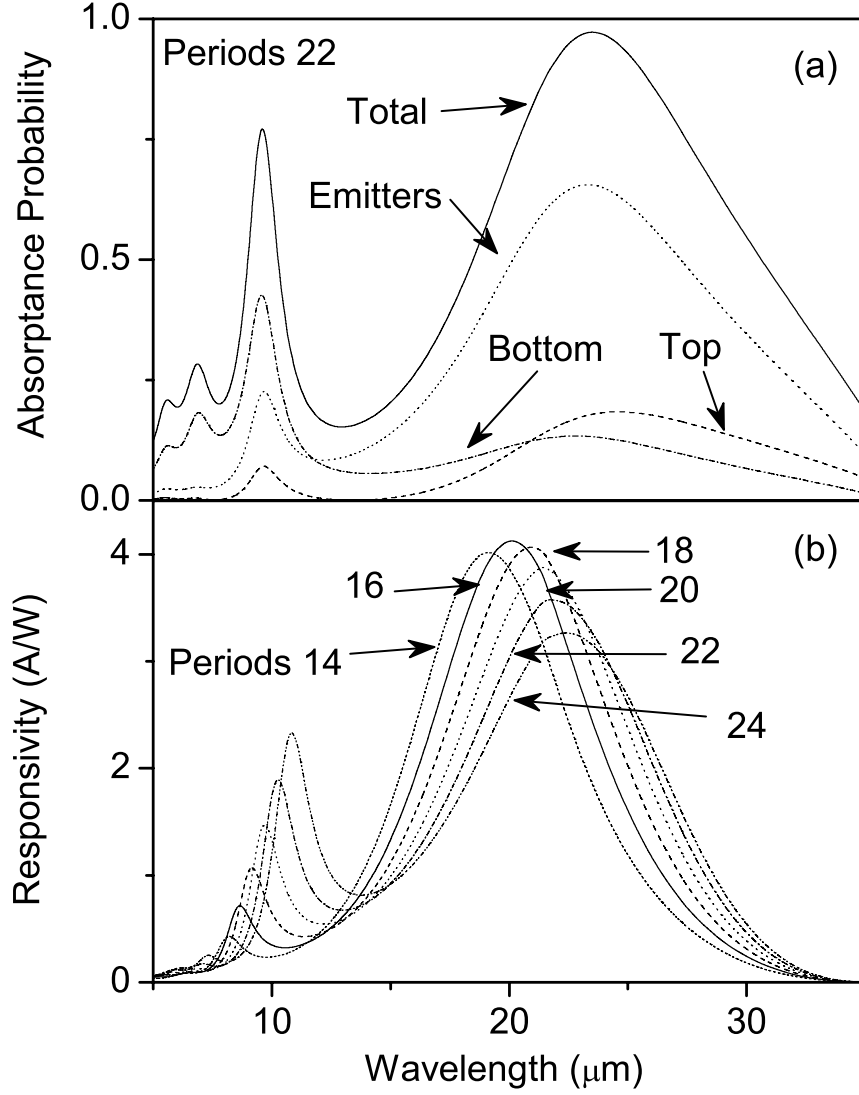


Figure 2.21: (a) Calculated total absorption in the top and bottom contacts, and in the emitters for structure IV with 22 emitter/barrier units. The absorption in the emitters contribute to both the forward and reverse directions. (b) The responsivity spectra for the same structure as a function of emitter/barrier units. The structure consists of a p -type top contact and a n -type bottom contact with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$. Emitters are p -type GaAs with a doping density of $1 \times 10^{19} \text{ cm}^{-3}$. The threshold wavelength, $\lambda_0 = 35 \mu\text{m}$.

reduces with the thickness. This is mainly due to the decrease in internal photoemission probability as the peak moves closer to λ_0 .

The thickness of the bottom contact is fixed approximately at three times the skin depth corresponding to the peak wavelength. A thinner bottom contact would be transparent and as a result, would decrease the effective absorption in the emitters. This will reduce the detector responsivity. As the evanescent waves spread to about three times a wavelength's skin depth, increasing the bottom contact thickness more than needed would not be useful. Furthermore, thin structures are economical and easy to grow technologically. The increase in doping density decreases the skin depth and the effective total thickness of the structure, causing a blueshift of the peak wavelength. Hence, by increasing the number of layers, a tradeoff is achieved, and this would fix the peak wavelength as designed. This would bring the peak back within the photoemission maximum, thereby restoring the designed high responsivity of the detector. According to calculations, for a doping density of $2 \times 10^{19} \text{ cm}^{-3}$, the number of layers should be 25. As expected, responsivity would further increase by 6%. But technological limits may restrict the level of doping density.

The optimal active layer thickness (in other words the number of emitter/barrier units) for a detector operating in a spectral range from λ_1 to λ_2 can be estimated as follows. First order maximum intensity of the standing wave corresponding to λ_1 occurs at a distance $\lambda_1/4\tilde{n}$ from the reflecting surface (in many cases it is the top surface of the bottom contact) where \tilde{n} is the refractive index of the layer. Similarly, the first order maximum corresponding to the wavelength λ_2 occurs at a distance $\lambda_2/4\tilde{n}$ from the reflecting surface. The separation between these two points fixes the positions of the first and the last emitters and decides

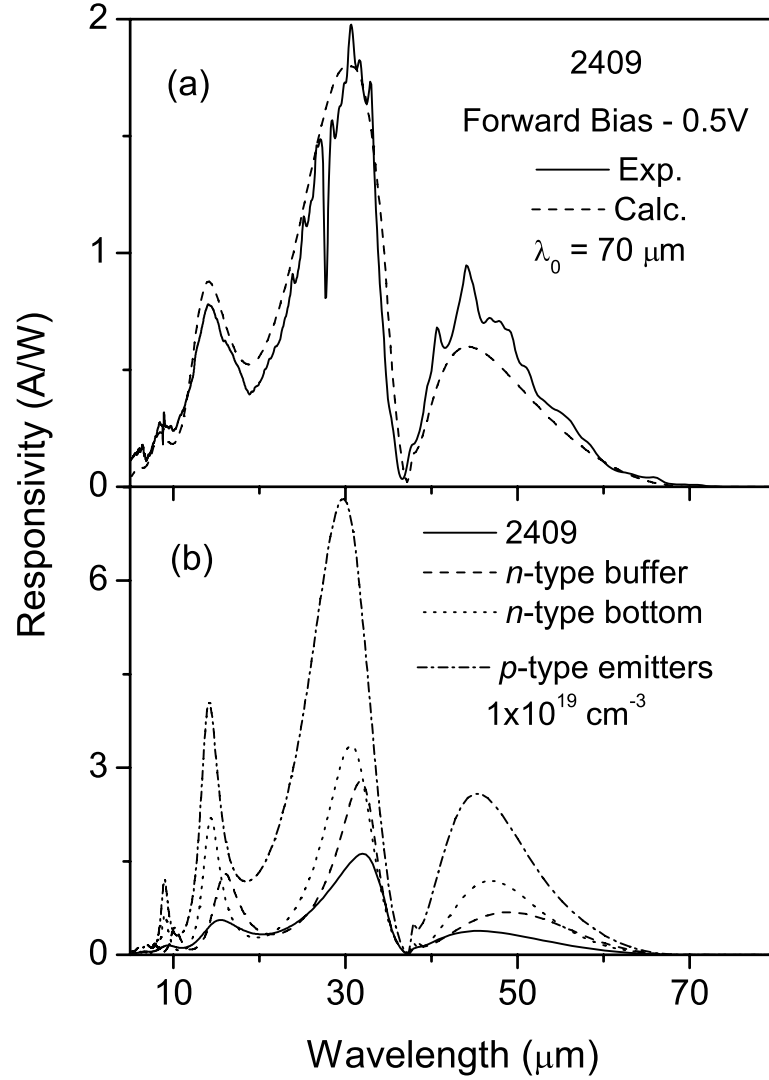


Figure 2.22: (a) Experimental and model responsivity spectra of detector 2409 with 30 emitter/barrier units. Doping density in the top and bottom contacts is $2.4 \times 10^{19} \text{ cm}^{-3}$, and the emitter has a doping density of $2 \times 10^{18} \text{ cm}^{-3}$. Thickness of the top contact, barrier layer, emitter and bottom contact are 208, 77, 15 and 730 nm, respectively. Total thickness of the structure is $3.5 \mu\text{m}$. The responsivity of an optimized 2409³⁰ structure is shown in (b). The responsivity increases by 1.7, 2.1 and 4.5 times if an additional n -type buffer layer, or a n -type bottom contact, or an increased emitter doping density is used, respectively.

the thickness of an optimal detector design.

The experimental and model responsivity spectra for a HEIWIP detector (detector 2409) are shown in Figure 2.22(a). This detector has 30 periods of emitter/barrier units giving 31 emitters (absorbers). Details of the parameters and experimental and model absorption spectra were published elsewhere³⁰. Calculations were performed in accordance with the model described above. The fitting parameter is a responsivity gain, which as defined has a value of $g \simeq 2$. The capture probability may be estimated as $p_c = 1/Ng \simeq 0.02$, which is close to the value for QWIP structures²⁶.

The model responsivity of a similar structure (to 2409) can be seen in Fig. 2.22(b). A n -type substrate with a doping density of 10^{18} cm^{-3} , serving as a mirror, reflects the incident radiation and increases the responsivity as a result. In practice, it is difficult to produce high quality substrates with doping density more than 10^{18} cm^{-3} . MBE or OMCVD technique permits the growth of doped epilayers up to a density of 10^{19} cm^{-3} . As shown in Fig. 2.22(b), the use of a $1.5 \mu\text{m}$ -thick n -type buffer layer between the p -type bottom contact and the substrate increases the responsivity by a factor of ~ 1.7 . While the use of a n -type bottom contact increases the responsivity by a factor of ~ 2 . The responsivity could be further increased (by a factor of ~ 4.5) upon increasing the doping density of emitters up to 10^{19} cm^{-3} . The calculation agrees well with the experimental results (see Fig. 2.22(a)), indicating that the model well describes the main features of the carrier generation and emission over the barrier, and could be used to design and optimize HEIWIP devices.

2.7 Detectivity and BLIP Temperature

In the background limited performance (BLIP) regime, the intrinsic noise of the detector is negligible compared to the noise arising from the incident background photon flux fluctuation. Hence the dark current is lower than the background photocurrent, and the total noise is determined by the photocurrent under the background illumination $\bar{I} = I_{photo}$. Under the detector limited condition, I_{noise} arises mainly from the detector, exactly from the fluctuation in the number of mobile carriers via generation-recombination (G-R) processes in the emitters, so $\bar{I} = I_{dark}$.

The photocurrent of the detector generated by a background flux of $T = T_{BG}$ and $FOV = 2\theta$ is given by

$$I_{photo} = \sin^2(\theta)A \int^{\lambda_0} R(\lambda)\rho(\lambda, T_{BG})d\lambda, \quad (2.36)$$

where $\rho(\lambda, T_{BG}) = 2\pi c^2 h / \lambda^5 (\exp(hc/kT_{BG}) - 1)^{-1}$ represents the blackbody distribution.

Experimental and model bias dependence of the BLIP temperature for detector HE0204²⁵ are presented in Fig. 2.23. The background temperature was $T_{BG} = 300$ K with a FOV of 62° . The experimental and calculated data are in good agreement for bias voltages lower than 3.5 V. The discrepancy at higher bias is due to the fact that the 3D drift model for the dark current in HEIWIP detectors is valid only for low electric field regime. Deviation of the dark current from calculation is observed for bias voltages above 2–3 V, as seen in Fig. 2.15.

Absorption probability in the structure (not taking the cavity effect into consideration) is approximately proportional to the total thickness, whereas the gain is inversely proportional to the same. This leads to a responsivity which is nearly independent of thick-

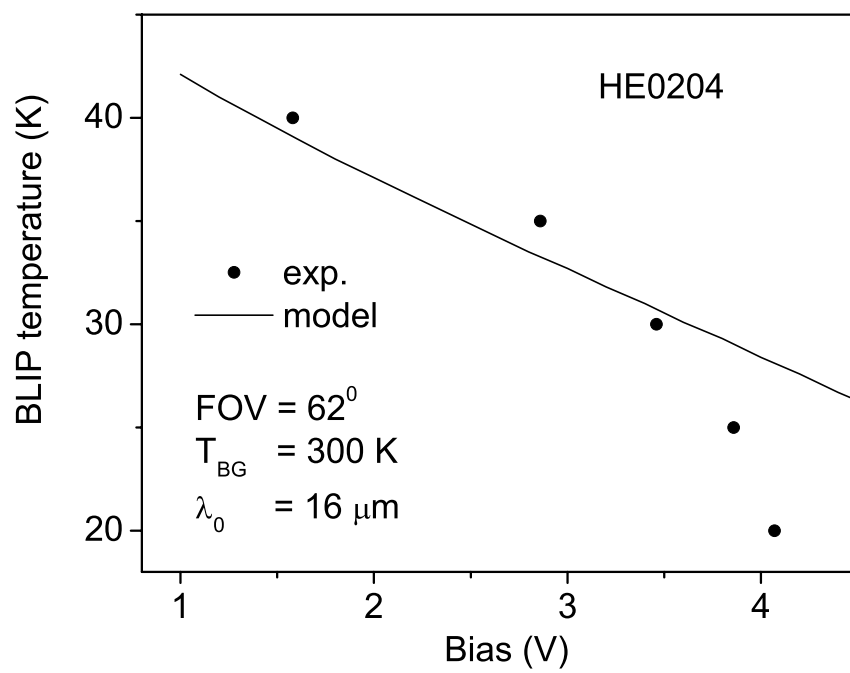


Figure 2.23: Experimental and model BLIP temperature vs. bias for detector HE0204. The background temperature was 300 K and the FOV of the detector at the cold stop was 62° .

ness and number of emitters. In the BLIP regime, detectivity is also independent of the thickness because the noise current depends only on the background fluctuations. But under the detector limited condition, the noise current given by Eq. 2.32 is proportional to the noise gain, which is inversely proportional to the thickness. So, at temperatures higher than T_{BLIP} , detectivity is higher for thicker structures due to lower gain and lower noise current. The detectivity, D^* , of the detector is defined as

$$D^* = \frac{R \cdot \sqrt{A}}{I_{\text{noise}}}, \quad (2.37)$$

where R is the responsivity, A is the electrical area, and I_{noise} is the noise current of the detector.

Model specific detectivity (D^*) spectra of HEIWIP detector 2409 in the BLIP regime are presented in Fig. 2.24. The calculated BLIP temperature was $T_{\text{BLIP}} = 13$ K for a background temperature of $T_{\text{BG}} = 300$ K and a FOV = 180°. The figure demonstrates a peak D^* of 1.7×10^{10} , 2.4×10^{10} , 2.5×10^{10} , or 3.4×10^{10} $\text{cm}\sqrt{\text{Hz}}/\text{W}$ for detector 2409, if an n-type buffer layer, n-type bottom contact, or highly doped p-type emitters are used, respectively. The D^* for an ideal detector with the same $\lambda_0 = 70$ μm is also shown in the figure.

2.8 Response Speed

There are three main mechanisms limiting the response time of HEIWIP detectors. These are the intrinsic relaxation time, the carrier transit time, and the RC time constant. The intrinsic time response of the $p\text{-GaAs}/\text{AlGaAs}$ HEIWIP detector can be estimated from the bias dependent responsivity measurements³¹. Under irradiation, excited carriers

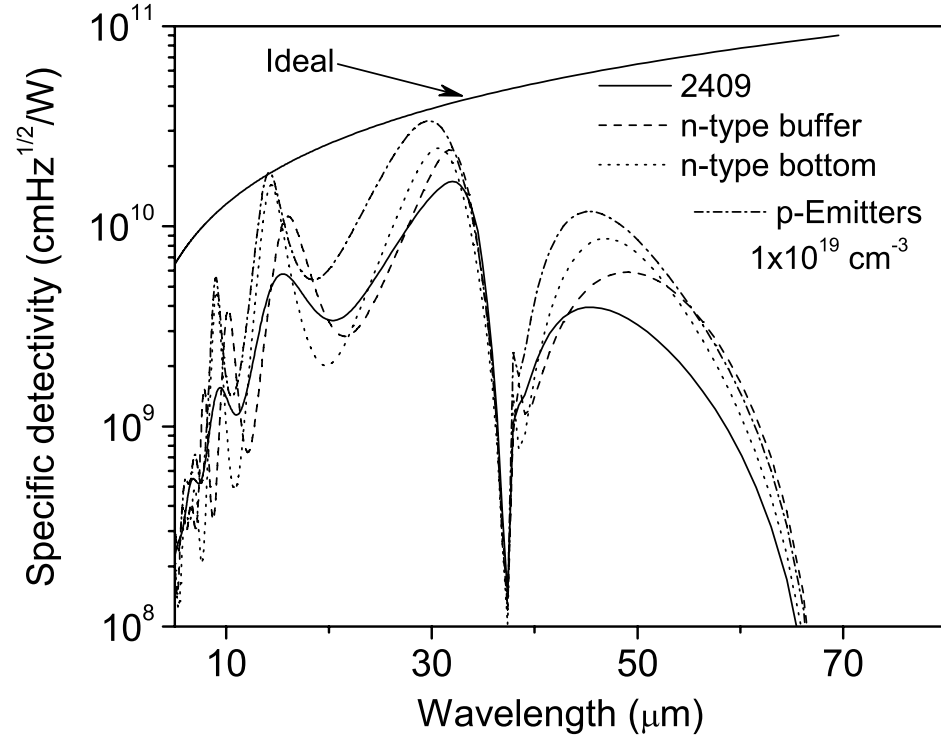


Figure 2.24: The solid line shows the model detectivity spectra for the 2409 HEIWIP detector in the BLIP regime (λ_0 is 70 μm). Background temperature is $T_{\text{BG}} = 300$ K and $\text{FOV} = 180^\circ$. The BLIP temperature, $T_{\text{BLIP}} = 13$ K. Model detectivities for structures similar to 2409, but with a 1.5 μm -thick n -type buffer layer doped to $1 \times 10^{19} \text{ cm}^{-3}$, or with a n -type bottom contact, or with the emitter doping density increased to $1 \times 10^{19} \text{ cm}^{-3}$ are shown for comparison. The top solid line represents the detectivity of an ideal detector with the same λ_0 .

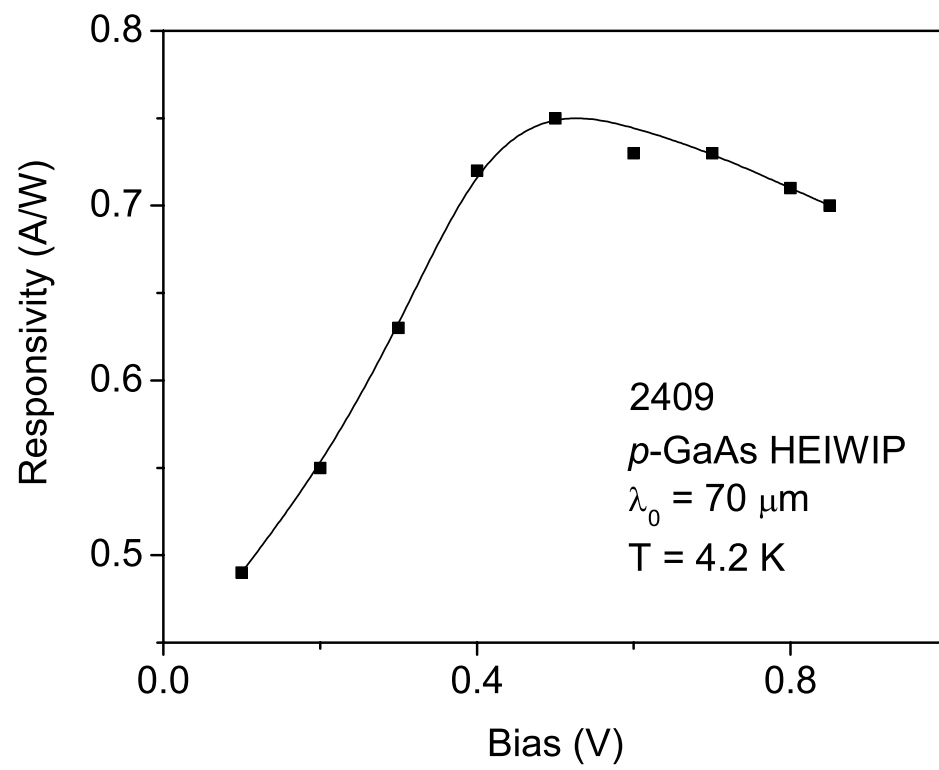


Figure 2.25: Bias dependence of responsivity in 2409 HEIWIP FIR detector with 30 emitter/barrier units; $\lambda_0 = 70 \mu\text{m}$.

are generated by free carrier absorption in the valance band. However, the total number of carriers remains constant with an effective temperature T_{eff} (which deviates from the equilibrium temperature, T_0) in a hot carrier population. The photoconductivity is given by:

$$\Delta\sigma = qp[\mu(T_{\text{eff}}) - \mu(T_0)] \quad (2.38)$$

where p is the carrier density, μ is the hole mobility. The change in conductivity under illumination leads to a change in the current in the external circuit (photocurrent) of $I_{\text{photo}} = \Delta\sigma FA$, where F is the applied field in the detector and A is its area. A minute heating of the carriers leads to:

$$\Delta\sigma = qp \left. \frac{d\mu}{dT} \right|_{T=T_0} \cdot (T_{\text{eff}} - T_0) \quad (2.39)$$

Therefore, the photoconductivity is directly related to the transport properties of the hot carriers. The temperature rise $T_{\text{eff}} - T_0$ has now to be related to the incident power P from the energy balance equation :

$$\frac{pk(T_{\text{eff}} - T_0)}{\tau} = \frac{P\eta}{Ad} \quad (2.40)$$

where k is the Boltzmann's constant, d is the detector thickness, τ is the energy relaxation time of the hot carriers, which in the limit, can be regarded as the detector response time. The left side term of Eq. 2.40 represents the power transferred to the lattice by hot carriers, and the right side term displays the power transferred to the hot carrier distribution. The current responsivity $R = I_{\text{photo}}/P$ under a bias voltage V can be expressed as:

$$R = \frac{q\eta\tau}{kd^2} \left. \frac{d\mu}{dT} \right|_{T=T_0} \cdot V \quad (2.41)$$

Using the temperature dependent mobility $\mu \propto T^{3/2}$, due to the ionized impurity scattering for the low experimental temperature of $T_0 = 4.2$ K, the current responsivity can be rewritten as:

$$R = \frac{3\eta\mu\tau}{2d^2} \frac{q}{kT_0} V \quad (2.42)$$

The bias dependence of responsivity measured at 4.2 K in a *p*-GaAs HEIWIP FIR detector with 30 periods and λ_0 around 70 μm is shown in Fig. 2.25. The measured responsivity increases linearly with the bias at low bias voltages as predicted by Eq. 2.42. The saturation of the responsivity at high bias is due to the quasi-depletion of the impurity band as proposed in a simple recombination model³². For a given bias, the responsivity is proportional to the response time, which is the same as in the case of an intrinsic or extrinsic photoconductive detector. The detector 2409 has a total absorption quantum efficiency of 0.07 at 14 μm , thickness of 3 μm , and a slope of 0.78 A/WV in the linear region of the R vs. V graph in Fig. 2.25. Using the above parameters the response time of this detector, determined by the energy relaxation time, is estimated to be about $\tau_{relax} \cong 6 \times 10^{-12}$ s. Assuming that the hole mobility to be $\mu = 60 \text{ cm}^2/\text{Vs}$, the transit time of the carrier through this structure, at an electric field of 3 kV/cm, was estimated to be $\tau_{trans} \cong 10^{-9}$ s.

In practice, the most serious limitation arises from the $\tau_{RC} = R_{total}C$ constant, where C is the capacitance of the detector and $R_{total} = R_L R_d / (R_L + R_d)$ is the equivalent resistance, where R_L is the load and R_d is the dynamic resistance of the detector. Typical values of R_d for a HEIWIP detector with a threshold wavelength $\lambda_0 = 70 \mu\text{m}$ are about 5×10^5 and $5 \times 10^4 \Omega$ for a bias of 0.5 and 0.7 V, respectively.

The capacitance of detectors having $400\mu\text{m} \times 400\mu\text{m}$ mesa size and 3 μm thickness

is about 6 pF. For a very high load resistance ($R_L \gg R_d$) the RC time constant is estimated to be 3×10^{-7} s. Increasing the detector size up to $800 \mu\text{m} \times 800 \mu\text{m}$ will increase the time constant up to 1.2×10^{-6} s. An estimation shows that the intrinsic time response of HEIWIP detectors is very fast, and in real situations, the detector time response is mainly restricted by the RC time constant of it.

Chapter 3

Homojunction Interfacial Workfunction Far Infrared Detectors

3.1 Introduction

The basic structure of a HIWIP detector²⁸ consists of a heavily doped emitter layer and a barrier layer sandwiched between the contact layers. For *p*-type structures, the interfacial workfunction, Δ , is the offset between the Fermi level of the emitter and the valance band edge of the barrier, arising due to band gap narrowing of the highly doped emitter layer. The detection mechanism involves free carrier absorption in the emitter layer, followed by the internal photoemission of photoexcited carriers across the interfacial barrier. These photoemitted carriers are swept out of the active region by the applied electric field and are

collected at the contact. Initially, it was believed that, in principle, GaAs FIR detectors could be designed with arbitrarily long threshold wavelength $[\lambda_0 = 1240 \text{ (meV } \mu\text{m})/\Delta]$ ³³ because Δ can be made arbitrarily small by increasing the doping density of the emitters³⁴. For example, it has been shown³⁵ that λ_0 was tunable from 76 to 85 μm by varying the Be:dopant density of the emitter layer from 1×10^{18} to $3 \times 10^{18} \text{ cm}^{-3}$. However, it was found that the optically induced transition of carriers from the heavy-hole (HH) to the light-hole (LH) band, in highly doped p -type emitters depletes states close to Fermi level of the emitter, thereby increasing the effective workfunction for photoemission³⁶. In other words, λ_0 of HIWIP detectors is limited within 100 μm . Therefore, it is important that p -type HIWIP detectors are designed to operate below the threshold limit set by the onset of free carrier state depletion through transitions from HH-hole to LH-hole band. The experimental results observed for three Carbon doped p -type GaAs HIWIP FIR detectors with different barrier thicknesses are presented in this chapter.

3.2 Experimental

The HIWIP detector structures, reported here, were grown using OMCVD technique on a SI-GaAs (100) substrate. During the growth, the substrate temperature was maintained at 610 °C. The structures consist of a bottom contact (p^{++}) layer, a barrier layer, an emitter (p^+) layer, and a top contact layer as shown in Fig. 3.1. The structures (namely, RU001, RU002 and RU003) were processed into mesas with different optical window areas for characterization purposes. The top contact and a part of the emitter layer were etched out, leaving about 800 Å-thick emitter regions in all the structures. The layer

Table 3.1: Main parameters of the three device structures as confirmed by SIMS. Labels N_{tc} (W_{tc}), N_{em} (W_{em}), N_b (W_b), and N_{bc} (W_{bc}) denote the doping density (thickness) of the top contact, emitter, barrier, and the bottom contact layers of the structures, respectively. After processing the devices, the etched-down thickness of the emitters is $\sim 800 \text{ \AA}$.

Sample	W_{tc} (nm)	$N_{tc} \times 10^{19}$ (cm^{-3})	W_{em} (nm)	$N_{em} \times 10^{19}$ (cm^{-3})	W_b (μm)	$N_b \times 10^{17}$ (cm^{-3})	W_{bc} (μm)	$N_{bc} \times 10^{19}$ (cm^{-3})
RU001	120	5.9	200	1.6	4.0	2.0	1.0	3.0
RU002	150	5.0	200	0.5	0.1	3.0	1.0	3.0
RU003	120	5.3	200	1.5	1.0	1.8	1.0	2.0

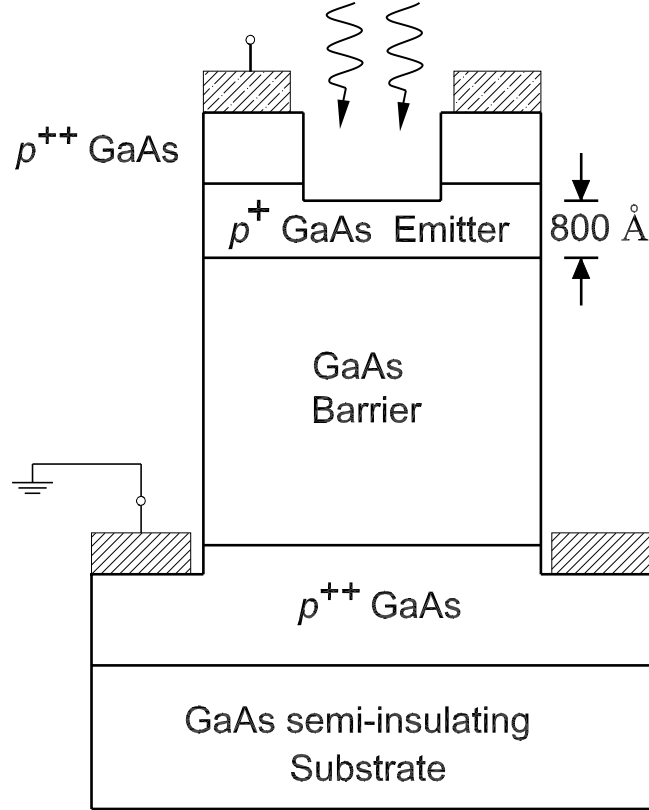


Figure 3.1: Schematic diagram of the p -type GaAs single emitter HIWIP detector after processing. A window is opened on the top for front illumination. The three structures RU001, RU002 and RU003 mainly differ in their barrier thickness, which is 4, 0.1 and $1 \mu\text{m}$, respectively. For all three detectors, the post-etched emitter thickness is $\sim 800 \text{ \AA}$, although the voltage drop still would apply through the unetched length of the device. The etching will increase the energy throughput to the emitter layer of the detector.

parameters (thickness and doping level) of the detector structures listed in Table 3.1 were confirmed by secondary ion mass spectroscopy (SIMS). It is noteworthy that the etched-emitter thickness of 800 Å was used for the photocarrier generation model, and the unetched emitter thickness was used for the photoemission model.

The detectors were characterized by current-voltage (I-V) and responsivity measurements. Transmission and reflection measurements were performed with a resolution of 4 cm^{-1} for the unetched samples. A Perkin-Elmer system 2000 Fourier transform infrared spectrometer (FTIR) with a Si composite bolometer as the reference was used for the spectral measurements. Transmission was measured for normal incidence, and the reflection was measured at an incidence angle of $\sim 8^\circ$. The absorptance spectra were obtained from the difference between unity and the sum of the reflectance and transmittance spectra³⁷.

3.3 Results and Discussion

3.3.1 Dark Current

The dark current of the detectors at $T = 4.2 \text{ K}$ is shown in Fig. 3.2(a). The inset shows the dark current in the narrow range of the electric field from -0.2 to 0.2 kV/cm. The sharp increase in the dark current for RU001 at an electric field of $\sim 0.1 \text{ kV/cm}$ could be attributed to tunneling associated with the material defects³⁸. Defect formation increases for thicker layers even for low defect materials such as GaAs³⁹. However, the dark current in the bias region of operation is still low and the performance of the detector is not affected.

The Arrhenius plots to determine the interfacial workfunction under forward bias fields of 0.05, 3, and 1 kV/cm for detectors RU001, RU002 and RU003, respectively, are

shown in Fig. 3.2(b). As shown in Fig. 3.2(c), the activation energy obtained for RU002 gives a barrier height of 14 meV for forward bias whereas the reverse gives a barrier height of 17 meV. For detectors RU001 and RU003, no difference was observed in the barrier height for forward and reverse bias. As listed in Table 3.1, the difference between the doping density of the emitter layer ($0.5 \times 10^{19} \text{ cm}^{-3}$) and the bottom contact layer ($3 \times 10^{19} \text{ cm}^{-3}$) is very large for detector RU002. This produces a built-in electric field in the barrier that was estimated to be about 2 kV/cm. The barrier lowering due to this internal field is ~ 5 meV at the emitter-barrier interface, which produces the different interfacial workfunctions for photoemission from the forward and reverse directions.

Dependence of the effective barrier height of the detectors on their applied field is shown in Fig. 3.3. Detector RU001 has a sharp drop in its barrier height at a field strength of 0.1 kV/cm, thereby limiting the operating field; this is directly related to the increasing dark current discussed before. Detectors RU002 and RU003 also show a slow barrier lowering due to image force. Therefore, their spectral λ_0 was expected to change with the applied field²⁸.

3.3.2 Detector Response

Room temperature experimental and model absorption spectra for unetched pieces, from the same wafers as the detector structures RU001, RU002, and RU003, are shown in Fig. 3.4. The model calculations were based on Eq. 2.18. Here, a two component free-carrier plasma consisting of heavy holes (HH) and light holes (LH) was considered. Although the light hole constitutes $\sim 5\%$ of the total free carrier density, the high mass ratio (7:1) of HH to LH reduces the plasma frequency of both to be of the same order⁴⁰. Details of

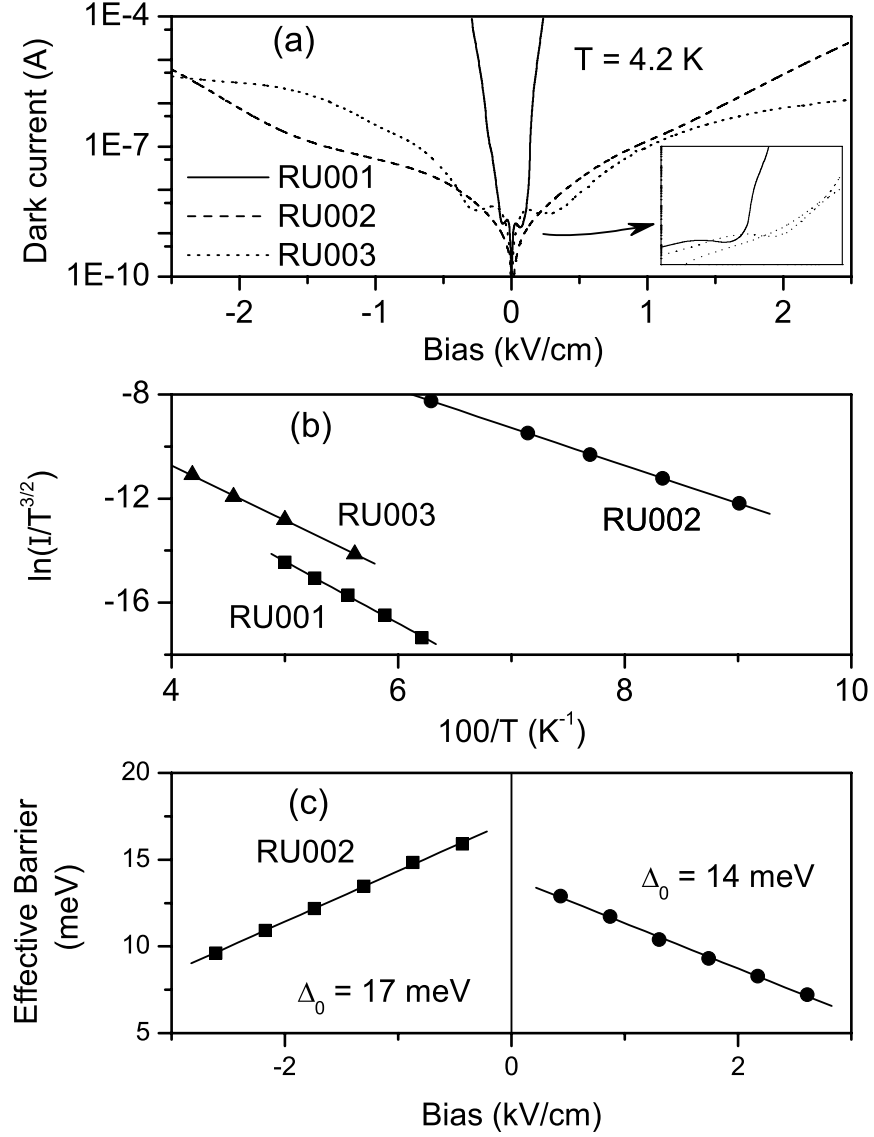


Figure 3.2: (a) Dark current curves for the three detectors at 4.2 K. The asymmetry in the dark current curves is due to non-uniformity in the structures. The rapid rise of dark current in RU001 can be attributed to defects in the barrier. The zoomed portion shown in the inset clearly shows the deviations. (b) Arrhenius plots under forward bias fields of 0.05, 3, and 1 kV/cm for detectors RU001, RU002 and RU003, respectively. (c) The Arrhenius plot for RU002 shows different interfacial workfunction in the forward and reverse direction. This is due to barrier lowering arising from asymmetry.

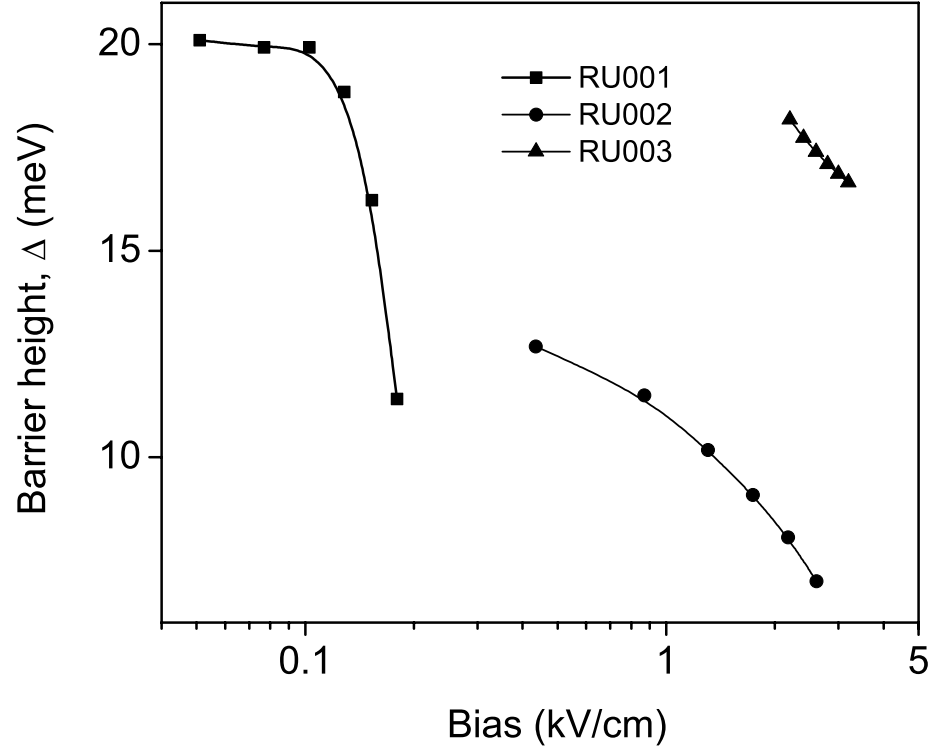


Figure 3.3: Variations of workfunction obtained through Arrhenious plots with the applied field. Detector RU001, with the 4 μm -thick barrier layer, has an almost constant workfunction up to 0.1 kV/cm, decreasing sharply afterwards. Whereas, the other two show the expected bias variation due to image force lowering. The deviation in RU001 explains the sharp increase observed in its dark current (see Fig. 3.2).

the dispersion model and calculations of the optical electric field distribution across the structure, the reflectance, and the transmittance were given in Chapter 2. Thickness and the doping level of the structures were obtained by fitting the model reflection spectra to the experimental results. These values agree within 5% of the design and SIMS data.

As the emitter and bottom contact layers of these detectors are highly doped, an enhancement³⁷ in the response of selective spectral regions, corresponding to different orders derived from detector's cavity-length, was expected. These wavelengths, around which such selective enhancement occurs, are approximated by:

$$\sum_j \text{Re}(\tilde{n}_j(\lambda))d_j = m\lambda/4, \quad m = 1, 3, \dots \quad (3.1)$$

where $\text{Re}(\tilde{n}_j)$ is the real component of the complex refractive index of the j^{th} layer and d_j is its thickness. The summation is carried throughout the layers down to the top of the bottom contact of the detector structure.

For structure RU001, taking the total thickness of the structure and the mean refractive index into account, the first ($m = 1$) and the third ($m = 3$) order resonance peaks were expected to be at $\lambda = 68$ and $23 \mu\text{m}$, respectively. The absorption peak corresponding to the first order resonance (indicated by an arrow on the model curve at $\lambda = 68 \mu\text{m}$) occurs around $\lambda \simeq 60 \mu\text{m}$ as shown in Fig. 3.4, and the peaks corresponding to higher orders fall outside the spectral range of measurement. Similarly, the first order peaks for structures RU002 and RU003 which were expected at $\lambda = 14.5$ and $26 \mu\text{m}$, respectively, and higher orders fall outside the range of measurements. The sharp drop around $37 \mu\text{m}$, the reststrahlen band for GaAs, is due to strong reflection caused by photon-optical phonon interactions.

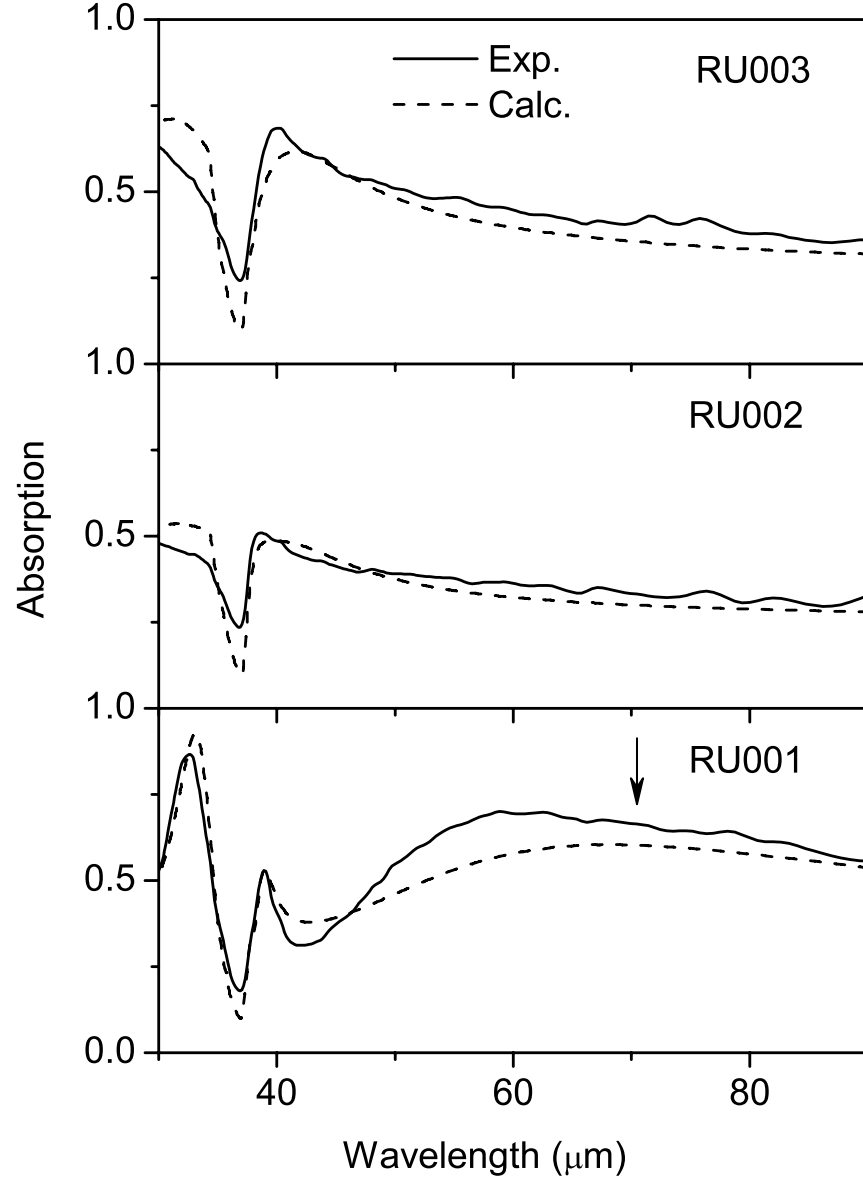


Figure 3.4: The experimental (solid line) and model (dashed line) absorption spectra for the structures with different barrier thickness at room temperature. The absorption measurements were done for pieces without etched top contacts. RU001 has a 4 μm -thick barrier layer whereas RU002 and RU003 have 0.1 and 1 μm -thick barriers, respectively. The first order cavity peak for structure RU001 was expected around the position, $\lambda = 68 \mu\text{m}$, shown by the arrow. Higher order peaks for structure RU001, and peaks of all orders for structures RU002 and RU003 fall outside the measured spectral range.

The responsivity of these HIWIP detectors showed a strong bias dependence as shown in Figs. 3.5 and 3.9. The maximum responsivity for detectors RU001, RU002, and RU003 was obtained at the bias fields of 0.05, 2.0 and 1.25 kV/cm, respectively, in the forward direction. Whereas, in the reverse direction it was obtained at the bias fields of 0.12, 2.0 and 2.5 kV/cm, respectively. As expected from dark current behavior, the performance of detector RU001 decreases as the field approaches 0.1 kV/cm. This decrease can be attributed to the high density of defects present in RU001's thick barrier region. The peak observed around $\lambda = 68 \mu\text{m}$ in the absorption spectra of structure RU001 (Fig. 3.4) corresponds to the first order cavity resonance. However, this was not observed in the responsivity spectra due to the suppression of photoemission of carriers excited by photons around the threshold wavelength, $\lambda_0 = 70 \mu\text{m}$.

Detector RU001 has a peak responsivity at $\lambda = 34 \mu\text{m}$, and its figures of merit are $R_p = 3.3 \text{ A/W}$, $D_p^* = 1.6 \times 10^{11} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, and $\eta_p = 12\%$ for a forward bias of 0.05 kV/cm. For a reverse bias field of 0.12 kV/cm, its figures of merit are $R_p = 3.5 \text{ A/W}$, $D_p^* = 1.7 \times 10^{11} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, $\eta_p = 12.8\%$. Even though this detector has a $4 \mu\text{m}$ -thick barrier layer, the $1 \mu\text{m}$ -thick bottom contact compared to the 800 \AA -thick emitter layer maintains a high carrier generation rate in the bottom contact layer. This results in the same order of responsivity observed for both bias directions. This detector can be used as a wide band detector operating in the range $40\text{--}65 \mu\text{m}$ with an average responsivity of 0.7 A/W .

The responsivity for the structure with $1 \mu\text{m}$ -thick barrier region (RU003) has a strong bias dependence, increasing significantly with the bias. However, the bias cannot increase indefinitely as the dark current also increases with the bias. At $\lambda = 34 \mu\text{m}$, it

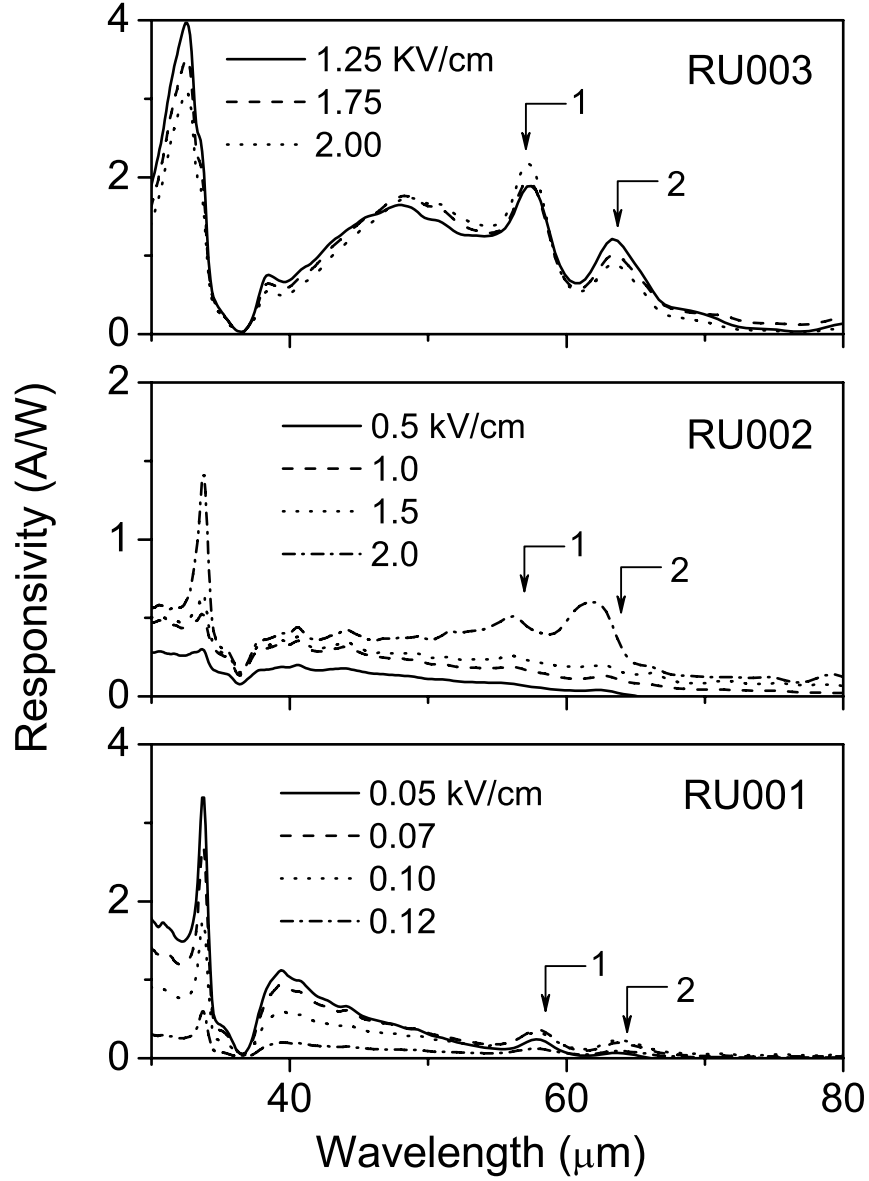


Figure 3.5: The experimental responsivity spectra for detectors at 4.2 K under different forward electric fields. The maximum responsivity observed was at $\lambda = 34 \mu\text{m}$, and it is 3.3, 1.4, and 4 A/W for detectors RU001, RU002 and RU003, respectively. The first order cavity peak for detector RU001 is around the threshold wavelength. The sharp drop around $37 \mu\text{m}$ is due to the high reflection in the reststrahlen band. Arrows 1 and 2 indicate the transitions between the ground and the excited impurity (carbon) states. Arrow 1: transition $1S_{3/2}(\Gamma_8) \rightarrow 2P_{5/2}(\Gamma_7)$, arrow 2: transition $1S_{3/2}(\Gamma_8) \rightarrow 2P_{5/2}(\Gamma_8)$ ⁴¹.

has a peak responsivity, R_P , of 7.4 A/W (4 A/W) giving a quantum efficiency of 27.2% (14.7%) and a detectivity, D^* , of 3.6×10^{11} cm $\sqrt{\text{Hz}}$ /W (1.9×10^{11} cm $\sqrt{\text{Hz}}$ /W) for 0.15 V/ μm reverse (0.125 V/ μm forward) bias. This detector shows a broad spectrum with an average responsivity of 3 A/W in the range 40–60 μm .

Peaks in responsivity were observed in both forward and reverse bias spectra at wavelengths of 57 and 63 μm for all three HIWIP detectors. They are due to hole transitions from the ground state to the excited states in the hydrogen-*like* coulomb wells of impurity atoms in the barrier. According to SIMS, the intentionally undoped barrier has become relatively low-doped due to carbon (C) migration from the highly doped sandwiching layers. It has been shown, from photoconductivity measurements of C impurities in GaAs, that the wavelength corresponding to transitions from the ground to the third excited impurity state ($1S_{3/2}(\Gamma_8) \rightarrow 2P_{5/2}(\Gamma_7)$) is 58.2 μm and from the ground to the second excited state ($1S_{3/2}(\Gamma_8) \rightarrow 2P_{5/2}(\Gamma_8)$) is 63.9 μm ⁴¹. Due to spin-orbit interactions, the degenerate state $2P_{5/2}$ splits into two states, $2P_{5/2}(\Gamma_7)$ and $2P_{5/2}(\Gamma_8)$, where point group symmetry for zinc-blende structures (like of GaAs) provide the notations Γ_7 and Γ_8 ⁴². The wavelength corresponding to transitions from ground state to the first excited state is 81 μm and lies outside λ_0 .

At low temperatures (T), the carriers have insufficient thermal energy ($k_B T$) to occupy the excited states, leaving the ground state almost fully occupied. This allows photoionization of holes filled in the ground state. Therefore, the photocurrent enhancement corresponding to transition wavelengths is due to photoexcitation and subsequent tunneling of carriers through the coulomb barrier. Hence, increasing the bias would decrease the effec-

tive tunneling width, thereby increasing the rate of photoionized carriers drifting towards the contact. The strength of the correlation with the applied electric field is expected to increase for the farthest (in terms of energy difference) transition. As shown in Figs. 3.5 and 3.9, the strongest transitions were observed at the highest field. The small deviation in the peaks for detectors RU002 and RU003 may be the result of Stark shifts of the impurity levels.

Increased responsivity in the 40–50 μm region, due to transitions from the impurity ground state to the excited state, is shown in Figs. 3.5 and 3.9. The ground state of the C-impurity atom lies at 26.9 meV⁴¹ above the edge of the valance band; therefore, the transition from the ground state of impurity to the continuum in the valance band was expected around 46 μm . Transitions from the ground state to higher excited states (1S→3P) and from the ground state to continuum, in the range 40–50 μm , have also been observed previously⁴¹. Because the ground state of the impurity wells are almost fully occupied, response due to impurity photoionization is more pronounced at low temperatures.

Peak responsivity, quantum efficiency, and detectivity of these single emitter detectors RU001, RU002, and RU003 were compared with another detector (9604). This previously reported detector has 20 periods of emitter/barrier units and the total thickness of the emitters is 0.3 μm . The figures of merit for the forward bias are listed in Table 3.2, which shows that the performance of these single emitter HIWIP is very good⁴³.

The performance of the detectors (responsivity, quantum efficiency, and detectivity) around the impurity transition regions are presented in Table 3.3, indicating the highest responsivity for detector RU003. The low responsivity in RU001 is due to the high density

of defects present in the structure preventing the operation at high bias fields comparable to others. Detector RU002 has the lowest number of impurities due to its thin barrier region. As a result, it has the lowest contribution to responsivity from impurity photoionization in its barrier.

The energy of plasma oscillations for the $5 \times 10^{18} \text{ cm}^{-3}$ doped emitter layer of detector RU002 is $\hbar\omega = 33.6 \text{ meV}$, and the optical phonon energies for GaAs are: $\hbar\omega_{TO} = 33.2 \text{ meV}$ and $\hbar\omega_{LO} = 36.1 \text{ meV}$. This produces a strong coupling between plasma oscillations and polar lattice vibrations. This gives rise to a plasmon with high damping, re-normalizing the plasmon frequency⁴⁴ and changing the free carrier absorption mechanism. However, the plasma frequencies for detectors RU001 and RU003 are 60.2 and 58.3 meV, respectively. This relatively large differences between the plasma and phonon frequencies for these detectors result in a weaker phonon-plasmon coupling.

The model and experimental response spectra for $T = 4.2 \text{ K}$ under forward and reverse bias are shown in Figs. 3.6 and 3.7, respectively. The comparison with calculations was done for low-bias experimental results because the model does not include any electric field effects. The calculations were carried out using structure parameters (thicknesses and doping densities) obtained by fitting the experimental reflectance to its model. In the response model, the thickness of emitter was taken to be 800 \AA as this is the effective post-etch thickness that remains for photocarrier generation. Although there is a reasonable agreement between the experimental and model curves (see Figs. 3.6 and 3.7), still the model significantly deviates from data beyond the reststrahlen region. This is partly due to the fact that the model does not include responsivity enhancement arising from photoionization of

Table 3.2: Figures of merit (peak responsivity, quantum efficiency and specific detectivity) for the single emitter detectors RU001, RU002 and RU003 at $\lambda = 34 \mu\text{m}$ compared to the multi-emitter detector 9604. The thickness of the emitter (800 Å) and the bottom contact (1 μm) for all three single emitter detectors are the same. Detector 9604 has 20 periods of emitter/barrier units (total thickness of the emitters is 0.3 μm) and its peak responsivity is at 34 μm .

Sample	Peak responsivity (A/W)		Quantum efficiency (%)		Specific detectivity $\times 10^{11}$ (cm $\sqrt{\text{Hz}}$ /W)	
	Forward	Reverse	Forward	Reverse	Forward	Reverse
RU001	3.3	3.5	12.0	12.8	1.6	1.7
RU002	1.4	3.6	5.0	13.3	0.7	1.7
RU003	4.0	7.4	14.7	27.2	1.9	3.6
9604	3.1	–	12.5	–	0.5	–

Table 3.3: Figures of merit (peak responsivity, quantum efficiency and detectivity) for the single emitter detectors RU001, RU002, and RU003 at $\lambda = 58$ and 64 μm . These wavelengths correspond to transition of carriers from the ground to the third and to the second excited states of the impurity-well (carbon), respectively. The maximum responsivity is achieved for detector RU003, whereas high density of defects would not allow the electric field to be increased significantly for the detector RU001, and detector RU002 has the lowest barrier thickness resulting in the lowest impurity density.

Sample	Peak responsivity (A/W)				Quantum efficiency (%)				Specific detectivity $\times 10^{11}$ (cm $\sqrt{\text{Hz}}$ /W)			
	Forward		Reverse		Forward		Reverse		Forward		Reverse	
	58	64	58	64	58	64	58	64	58	64	58	64
	(μm)											
RU001	0.33	0.23	0.96	1.0	0.7	0.4	2.1	1.9	0.16	0.11	0.47	0.49
RU002	0.51	0.6	0.91	0.70	1.1	1.2	2.0	1.4	0.25	0.29	0.44	0.34
RU003	2.18	1.21	18.6	9.51	4.7	2.4	40.0	18.6	1.06	0.59	9.04	4.62

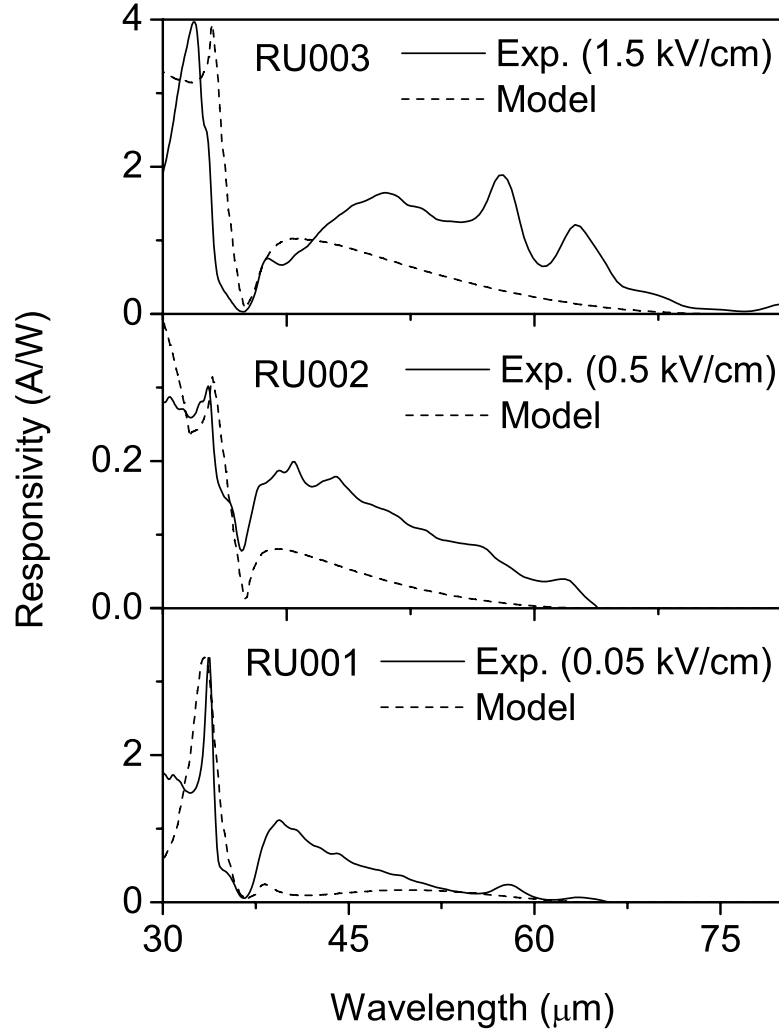


Figure 3.6: Model fit to experimental spectra of detectors RU001, RU002, and RU003 under forward bias at $T = 4.2$ K. The dip in the responsivity around $37 \mu\text{m}$ is due to high reflection in the reststrahlen band of GaAs. Unlike in other fits, the model deviates from the data significantly around the region where contribution to photoionization from the impurity atoms in the barrier is significant. The model used here does not include this mechanism.

impurity atoms in the barrier. More precisely, the peaks at 58 and 64 μm , and the increased responsivity in the range 40–50 μm in the experimental spectra are due to photoionization of holes, trapped in *H-like* wells formed by C impurities in the barrier, from ground to excited states and from ground to the valance band continuum. These mechanisms were not included in the theoretical model. The peak at ~ 34 μm and the sharp drop at ~ 37 μm are due to interaction of radiation with LO and TO optical phonons, and the variation of the optical electric field in the emitter and bottom contact, as discussed later.

As seen from Figs. 3.6 and 3.7, forward and reverse bias responsivities at low electric fields for detector RU002 are the lowest because it has the lowest optical electric field in the emitter. The sum of the tangential component of the optical electric field of the incident, transmitted and reflected waves must be zero at the surface of the highly conducting bottom contact. Therefore, it is understood that the optical electric field of any standing wave formed within the dielectric stack should increase with the stack thickness. It is reasonable that the strength of any optical field maximum formed in the emitter of RU002 is the lowest, compared to others (RU001 and RU003), for this has the lowest stack thickness. This gives a lower photon absorption probability (see Eq. 2.23), resulting in the weakest responsivity.

The above conclusion was confirmed by calculating the optical electric field distribution in the detector structures. The spectra of the normalized-mean-square optical electric field across the emitter and the bottom contact layers are shown in Fig. 3.8. The field for detector RU002 is the lowest compared to others. This, together with the lowest doping density in the emitter, leads to its responsivity being the lowest. The sharp drop

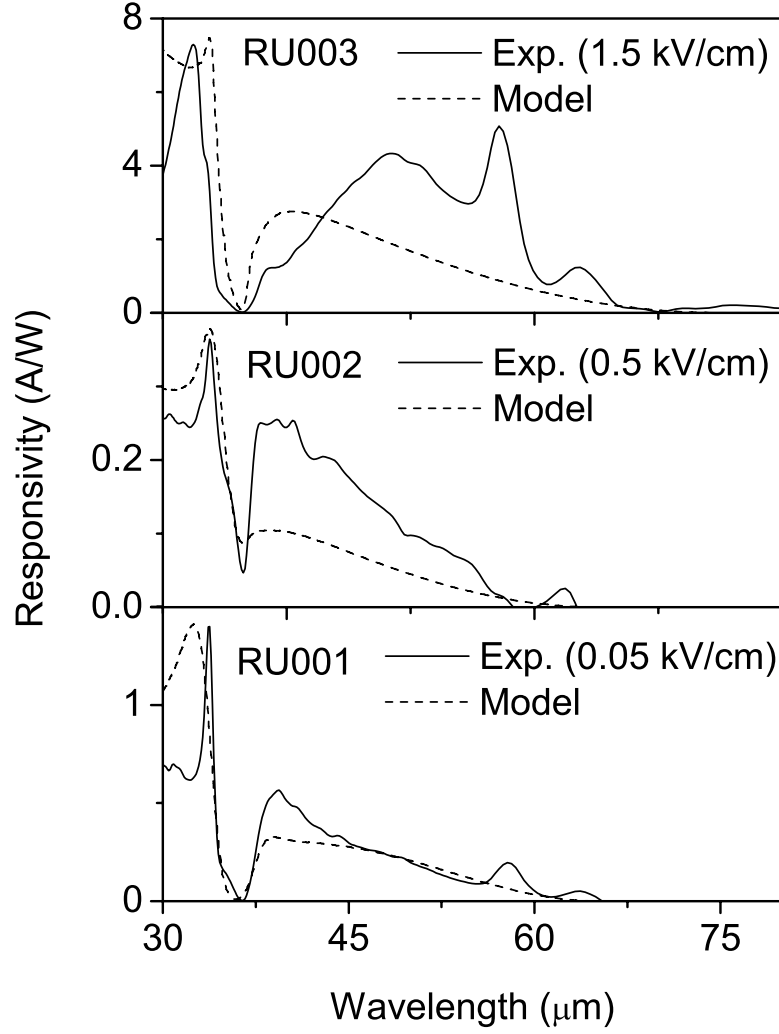


Figure 3.7: Responsivity vs. Wavelength for reverse bias at $T = 4.2$ K; solid lines: experimental data, dashed lines: model. The matching of model to data was done at electric fields 0.05, 0.5, and 1.5 kV/cm for detectors RU001, RU002 and RU003, respectively. The sharp dip in responsivity around $\sim 37 \mu\text{m}$ is due to strong interaction of photons with phonons in GaAs.

at $\lambda = 36.7 \mu\text{m}$ (where the permittivity shoots up) is due to GaAs-TO phonons. At the wavelength corresponding to LO phonons ($33.9 \mu\text{m}$), the permittivity is almost zero. This creates a high electric field, producing a responsivity peak in the experimental spectra as shown in Figs. 3.5 and 3.9. For detector RU001, the first order resonance cavity peak was observed at $\lambda = 68 \mu\text{m}$. The third order peak for this detector and the first order peak for detectors RU002 and RU003 fall outside the measured spectral range, as described earlier. The small peak shift, from $33.9 \mu\text{m}$ to shorter wavelengths, in detector RU001 is due to contribution from the strong third order resonant cavity peak at $\lambda \simeq 23 \mu\text{m}$. The optical electric field in the bottom contact is less than that in the emitter for all the detectors as shown in Fig. 3.8. The relative magnitudes of the responsivity for forward and reverse directions (photogeneration of holes in the emitter and the bottom contact, respectively) depend on the relative values of their optical electric fields, thicknesses, and the scattering lengths of the carriers.

The optical electric field distribution of the standing wave across the structures for $34 \mu\text{m}$ is shown in Fig. 3.10. The maximum intensity is obtained at the emitter surface and the field in the bottom contact is very low for all detectors. Detector RU002 has a relatively low electric field inside the barrier and at the surface, resulting in the lowest responsivity. Because the skin depth corresponding to $34 \mu\text{m}$ is greater than the thickness of the emitter and the bottom contact, the detector would allow the radiation to penetrate into the substrate.

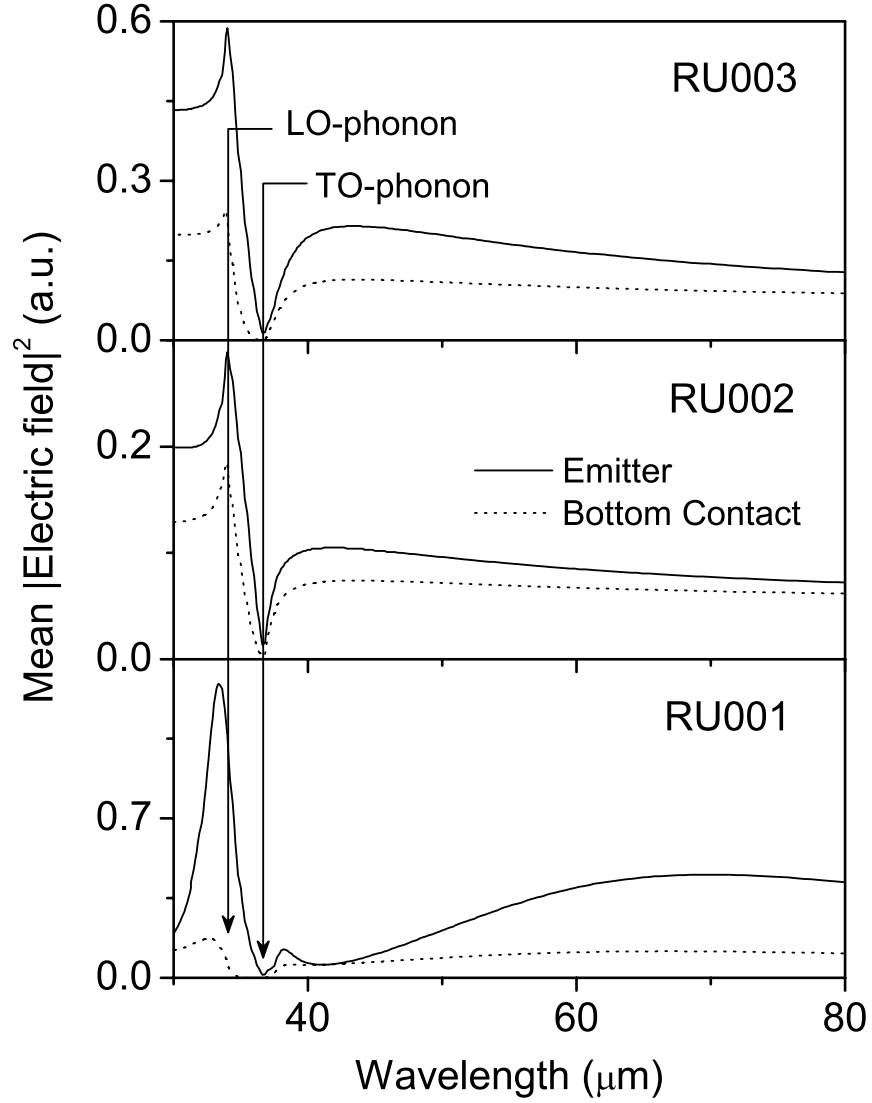


Figure 3.8: Calculated mean square of the optical electric field across the emitter and bottom contact layers of the three detectors. The lowest electric field produced the weakest responsivity for detector RU002. The sharp drop in the vicinity of GaAs-TO phonons ($36.7 \mu\text{m}$) is due to strong TO-phonon photon interaction. The peak electric field at LO phonon frequency ($33.9 \mu\text{m}$) is due to the dropping of the permittivity to zero. The drop and rise of electric field result in the drop and rise of the responsivity as in Figs. 3.5 and 3.9.

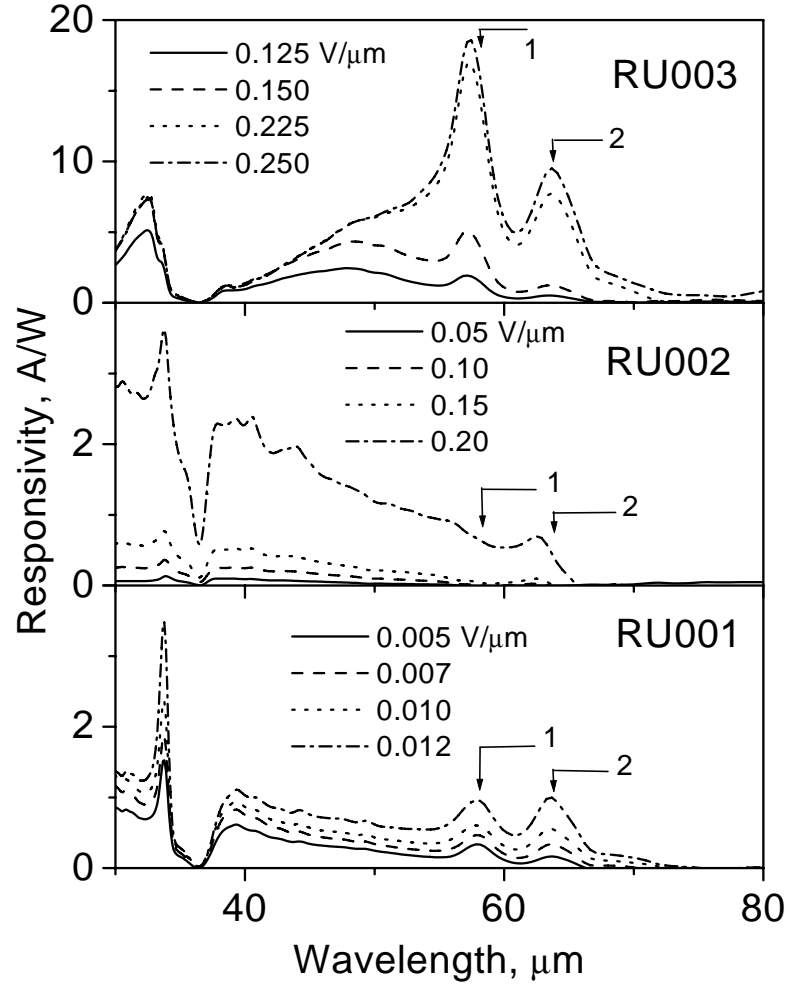


Figure 3.9: The experimental responsivity spectra for detectors at $T = 4.2$ K under reverse bias. Maximum responsivity is at $34 \mu\text{m}$, and its value is 3.5, 3.6 and 7.4 A/W for detectors RU001, RU002 and RU003, respectively. The first order cavity peak for detector RU001 is around the threshold wavelength. The sharp drop around $37 \mu\text{m}$ is due to the high reflection in the reststrahlen band. Arrows 1 and 2 indicate the transitions of holes between ground and excited states of the coulomb well formed by impurity (carbon) atoms in the barrier. Arrow 1: transition $1S_{3/2}(\Gamma_8) \rightarrow 2P_{5/2}(\Gamma_7)$, arrow 2: $1S_{3/2}(\Gamma_8) \rightarrow 2P_{5/2}(\Gamma_8)$ ⁴¹. The high responsivity at $58 \mu\text{m}$ for detector RU003 is due to high tunneling probability of photoexcited carriers at large electric fields (width of the tunnel barrier decreases with field), increasing the excited-carrier escape rate.

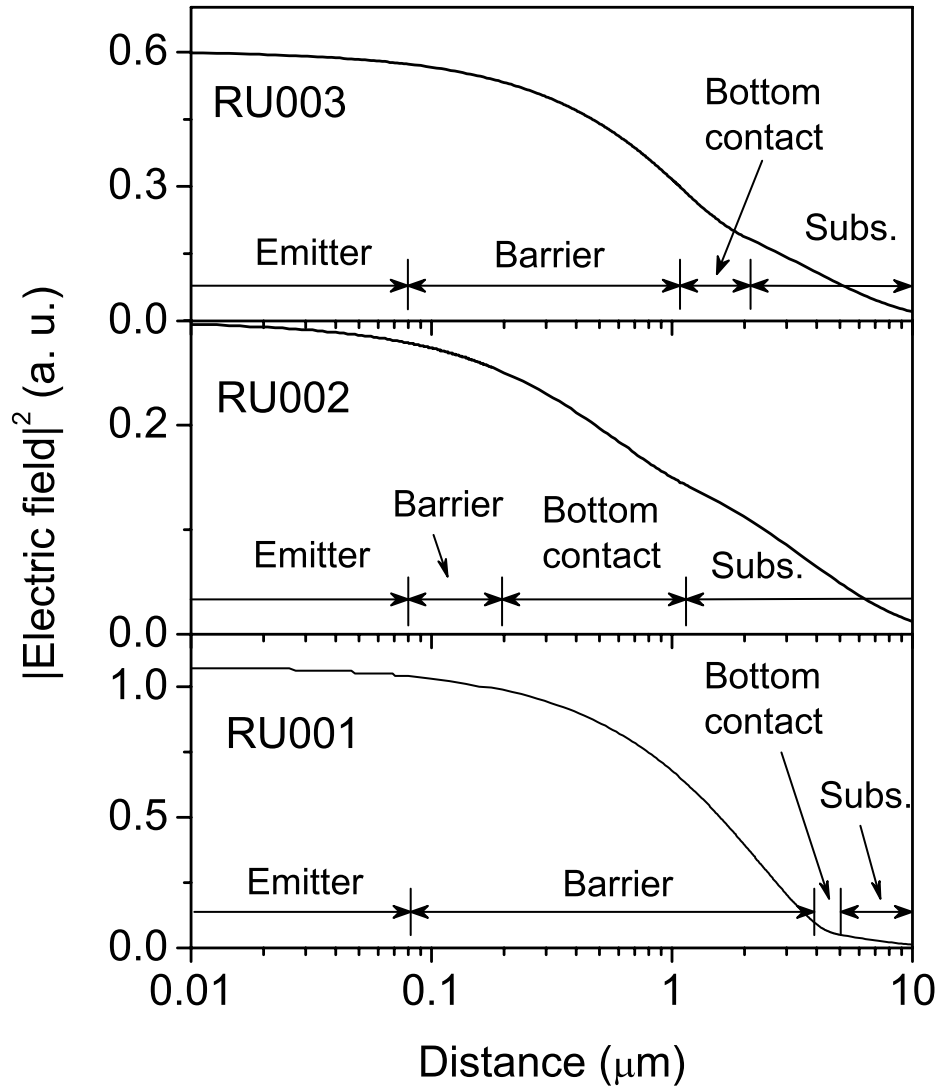


Figure 3.10: Calculated optical electric field distribution of the standing wave generated within the dielectric stacks for $\lambda = 34 \mu\text{m}$. The maximum field is on the surface of the emitter. The low field strength for detector RU002 is due to its stack-thickness being the lowest compared to RU001 and RU003.

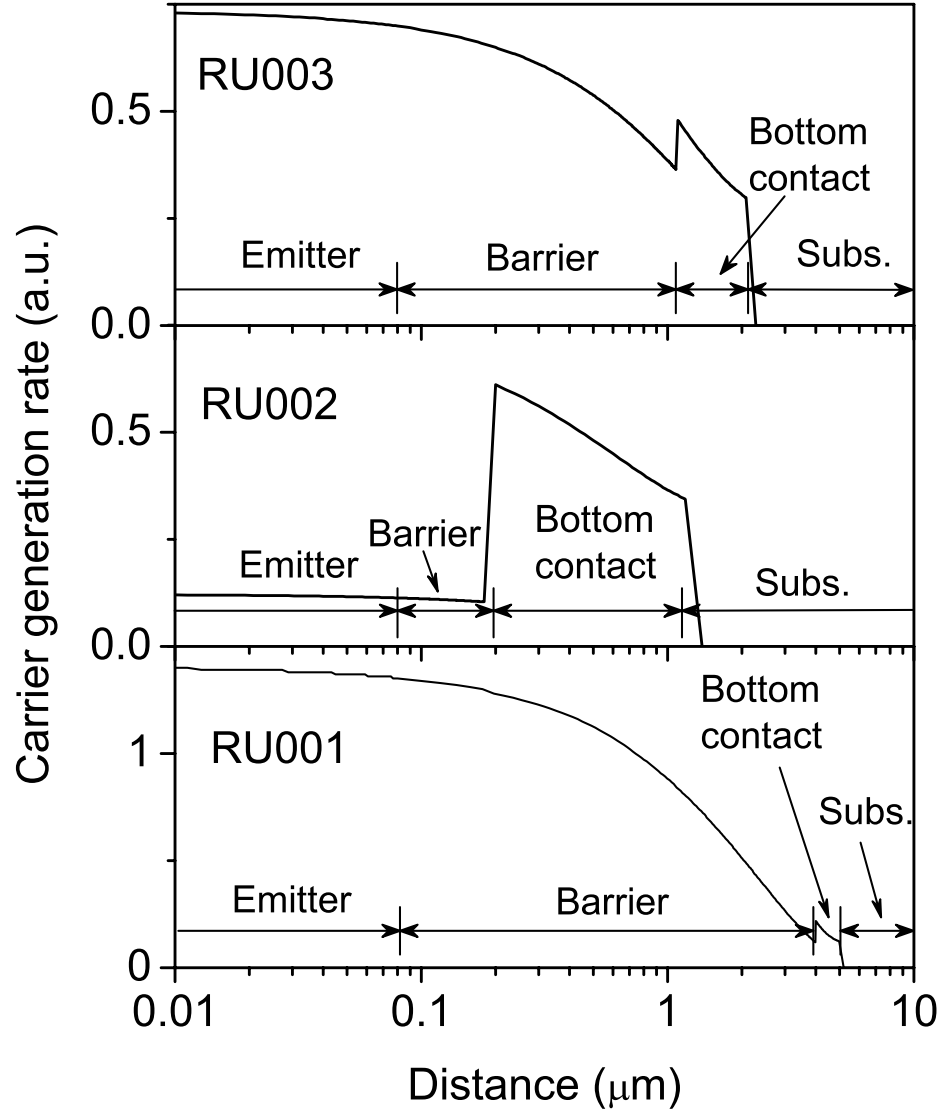


Figure 3.11: Calculated hot carrier generation rate across the structures for $\lambda = 34 \mu\text{m}$. High generation rate in the emitter and the bottom contact layers is due to their high doping density. For detector RU002, the carrier generation rate of the emitter layer is lower than that of the bottom contact layer.

Photon absorption probability in the layers not only depends on the electric field, but also on the imaginary part of the dielectric constant (see Eq. 2.23), which in turn depends on the doping density. The distribution of carrier generation rate, defined as the attenuation rate of η_a across the structure, is shown in Fig. 3.11 for $\lambda = 34 \mu\text{m}$. Sample RU002 has the lowest carrier generation rate in the emitter leading to the lowest responsivity. This is due to the weak optical electric field in the emitter and its low doping density. As shown in Fig. 3.11, in contrast to the emitter, the generation rate in the bottom contact is non-uniform, and this was taken into account in the responsivity calculation.

3.4 Conclusion

A high quantum efficiency with low dark current and an increased responsivity were observed for devices with 1, 0.1, and 4 μm -thick barrier layers. The dark current densities for these structures are on the order of 1–10 $\mu\text{A}/\text{cm}^2$ at $T = 4.2 \text{ K}$, corresponding to a high dynamic resistance compared with previous HIWIP FIR detectors. A detector with a barrier thickness of 1 μm had a peak responsivity of 18.6 A/W, a peak detectivity $D^* = 9 \times 10^{11} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, and a quantum efficiency of 40% at $\lambda = 58 \mu\text{m}$ under reverse bias. Threshold wavelengths of these detectors vary with bias and are around 70 μm as expected. Even though all three detectors are single emitter layer structures, the responsivity is greater than for the previously demonstrated multilayer structures. Responsivity and threshold wavelength show a strong bias dependence as expected in HIWIP detectors. Increase in the responsivity in the range 40–60 μm is due to photoionization of impurity atoms in the barrier.

Chapter 4

Heterojunction Interfacial Workfunction Far Infrared Detectors

Heterojunction interfacial workfunction internal photoemission detectors have proved their usefulness in many ways. These operate on the same detection principle as HIWIP detectors. For terahertz detection, HEIWIP detectors are an automatic choice as HIWIP detectors only operate in the region $\lambda < 70 \mu\text{m}$. As the doping density in the emitter is the only factor that decides the λ_0 for a given material, in theory, increasing the λ_0 beyond this limit necessarily requires emitters with doping density in the 10^{19} cm^{-3} region. Practically, however, such highly doped HIWIP detectors may not realize the designed λ_0 practically due to two major reasons. The first and foremost; a high doping density used in a *p*-type HIWIP detector could initiate the transitions of carriers from heavy- to light-hole band,

limiting the threshold wavelength to a maximum for the given material. Secondly, increasing the doping density in the emitter could increase the detector dark current. The most common p -dopant, Be, diffuses spontaneously at very high doping densities and could create defect centers in the sandwiching barriers. This could lead to defect assisted tunneling of carriers, increasing the detector dark current levels. Therefore, in one way or the other, increasing the doping would mostly result in defeating the purpose. Whereas, constructing the barrier with a heterojunction emitter/barrier unit will eliminate the need for high doping densities because the interfacial workfunction can also be changed through the alloy composition, generally, of the barrier layer. One of the side effects of this approach is the advantage of optimizing the doping density in the emitter to balance the dark current and the photon absorption efficiency while fixing the designed threshold.

The interfacial workfunction (Δ) in HEIWIP detectors has two contributions, the doping offset in the emitter (Δ_d) and the band offset arising from the alloy composition (Δ_x) in the barrier. These two offsets result in the workfunction as shown in Figs. 4.1 and 4.2. For a GaAs/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ HEIWIP, the latter offset, Δ_x , is calculated using a conduction band to valance band offset ratio of 65 to 35. At $T = 4.2$ K, the valance band offset varies as $530 \times x$, where x is the Al composition in the barrier. In p -type GaAs, the offset due to doping only varies from ~ 10 to 9 meV over a doping density change from 1×10^{18} to $1 \times 10^{19} \text{ cm}^{-3}$ (see Fig. 2.3(a)). The dashed lines on the flat band diagram on the right show the valance band edge ($k = 0$) of the barrier for $x = 0$. This corresponds to the barrier of a HIWIP detector with same doping. The Fermi energy of a n -type GaAs emitter is very large due to low *electron effective mass*. For example, for an emitter with $1 \times 10^{18} \text{ cm}^{-3}$

doping density, $E_F = 54$ meV, compared to a band gap narrowing of 34 meV. This will result in the Fermi level of the emitter being 20 meV above the $x = 0$ level. Therefore, in order to operate as a detector, this offset has to be overcome using the Al offset of barrier, which corresponds to an alloy fraction of ~ 2.5 %. Any additional Al fraction will then constitute the interfacial workfunction for the photoemission of the free carriers.

The variation of the threshold wavelength for both p and n -type GaAs/Al_{*x*}Ga_{1-*x*}As HEIWIP is shown in Fig. 4.3. For n -type, the threshold variation has a strong correlation with the Al fraction of the barrier, reaching $\lambda_0 \simeq 100$ μm for $x \simeq 3.5$, and for p -type x needs to be less than 0.5 % to reach this λ_0 . However, the aluminum fraction growth accuracy, and the transition from alloy to iso-electronic doping behavior in MBE grown structures define the lower limit of x , limiting the threshold to around ~ 100 μm .

4.1 Heterojunction Detectors Operating up to 20 μm

Detectors operating in the 8–20 μm range are attracting increased attention. Recent applications such as the transmission of digital signals using lasers with λ in the range 7–10 μm ⁴⁵ have been reported. The development of quantum cascade (QC) lasers operating at 21.5 and 24 μm ²⁹ will provide opportunities for extending communication applications to longer wavelengths which will require fast detectors operating at wavelengths longer than the 20 μm currently available⁴⁶ with mercury cadmium telluride (MCT) and QWIPs. To make the best use of these new long-wavelength lasers fast detectors operating in that specific range will be required. The wavelength range up to ~ 25 μm is particularly useful for studying molecular clouds and dust clouds near stars^{2, 3}. Direct measurements of tempera-

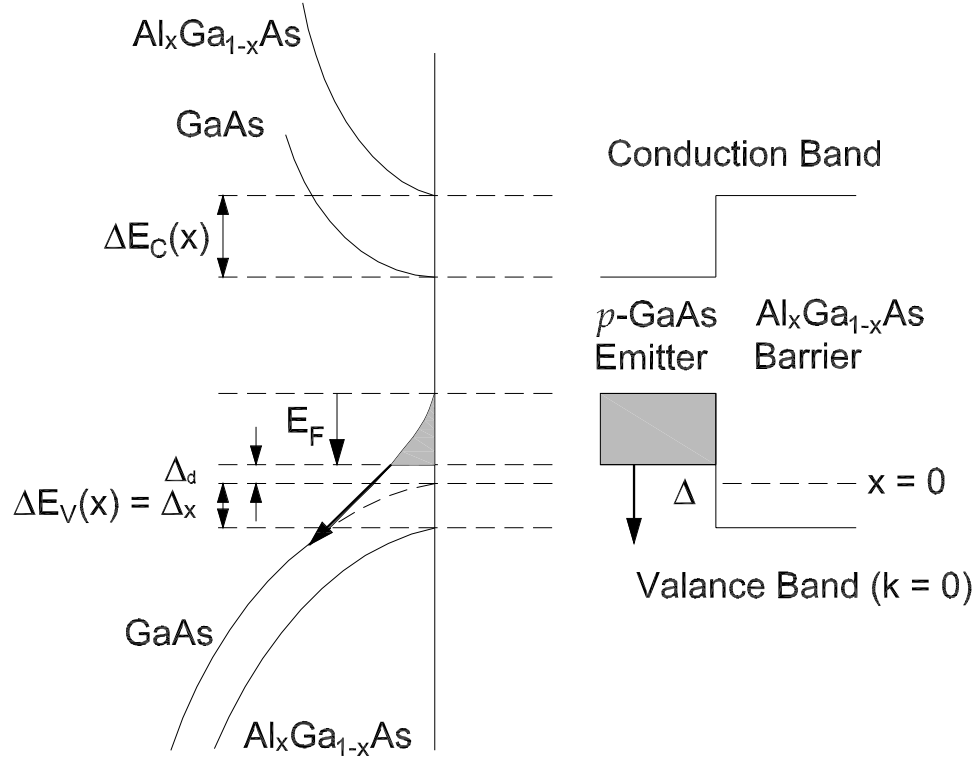


Figure 4.1: Band diagram showing the band gap narrowing and the Fermi level shift in a p -type doped GaAs emitter, and the band offset of the adjacent $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier. This is a type-I system where the bandgap of GaAs is completely within the band gap of $\text{Al}_x\text{Ga}_{1-x}\text{As}$. The conduction band to valance band offset ratio is $\sim 65:35$ of the total offset. Therefore, at $T = 4.2$ K, the valance band offset varies as $530 \times x$, where x is the Al composition in the barrier. In p -type GaAs, the offset due to doping only varies by ~ 1 meV for a doping density change from 1×10^{18} to $1 \times 10^{19} \text{ cm}^{-3}$ (see Fig. 2.3(a)). The interfacial workfunction (Δ) is the sum of the doping offset (Δ_d) of the emitter and the valance band offset (Δ_x) of the barrier. The dashed line on the flat band diagram on right shows the valance band edge ($k = 0$) of the barrier before offset ($x = 0$), resembling the interface in a HIWIP detector.

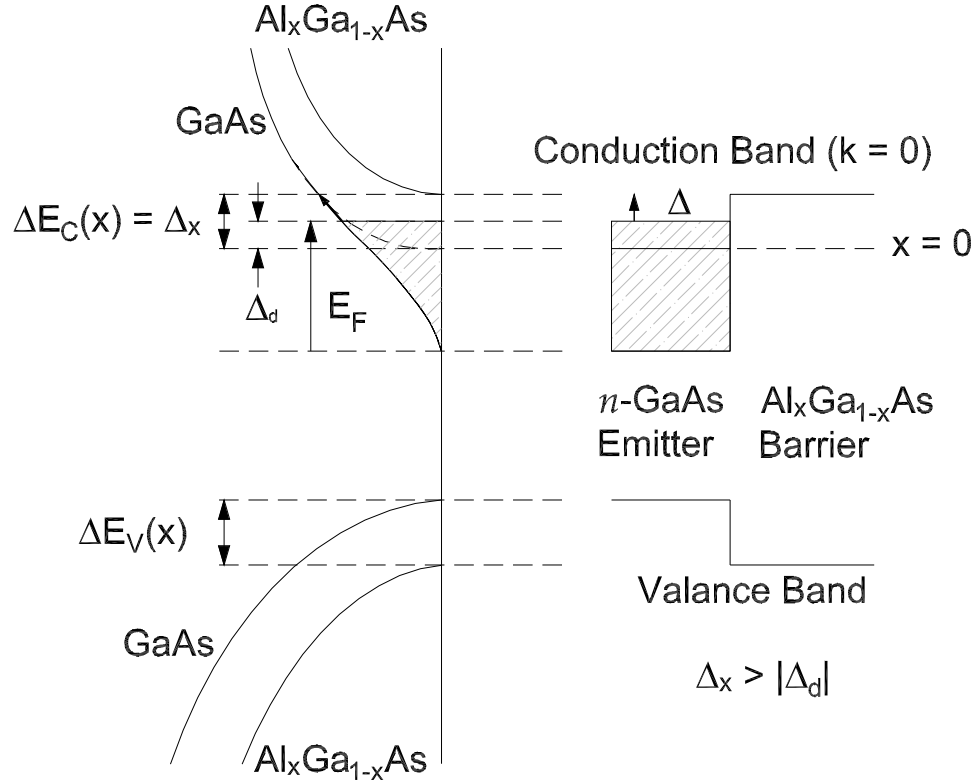


Figure 4.2: Band diagram showing the band gap narrowing and the Fermi level shift in a n -type doped GaAs emitter, and the band offset of the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier. Unlike in p -GaAs, the doping offset in n -type strongly depends on doping density. For a doping density of $1 \times 10^{19} \text{ cm}^{-3}$, the offset results in the Fermi level being above the barrier at $x = 0$. The interfacial workfunction $\Delta = \Delta_x + \Delta_d$, where $\Delta_d < 0$ for $n > 1 \times 10^{17} \text{ cm}^{-3}$. This negative offset should be taken into account in determining the alloy fraction of the barrier layer for a given interfacial workfunction. The dashed line on the flat band diagram on the right shows the valance band edge ($k = 0$) of the barrier before offset ($x = 0$).

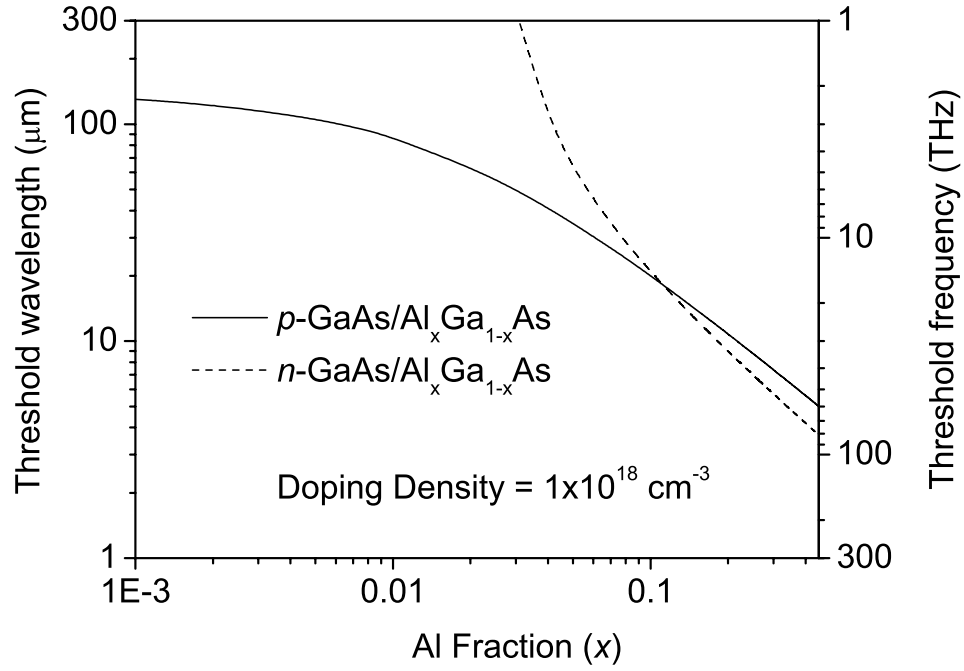


Figure 4.3: Threshold wavelength vs. Al fraction for p and n -type $\text{GaAs/Al}_x\text{Ga}_{1-x}\text{As}$ HEI-WIP detectors. The threshold wavelength for n -type detectors increases rapidly with decreasing alloy fraction. For p -type, the Al fraction required to increase λ_0 close to $100 \mu\text{m}$ is less than 1 % whereas for n -type it is ~ 3.5 %, which is relatively easy to control. The upper limit in Al fraction corresponds to a transition point beyond where the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier becomes indirect and photoexcited carriers need an additional phonon for emission. Meanwhile, for practical purposes, the lower limit is $x \simeq 0.005$. Aluminum fraction growth accuracy and the transition from alloy to iso-electronic doping behavior in MBE grown structures define the lower limit of x in HEI-WIP detectors.

ture and mass density can be obtained from the broadband absorption in dust and spectral lines in molecular hydrogen, water vapor, methane, and other molecules, resulting in mapping star formation regions and events like circumstellar shock waves. The wide range of applications in this range makes development of new quantum detectors of an immense interest.

4.1.1 Experimental

Three HEIWIP structures (labeled HE0204, HE0205 and HE0206) were grown using MBE technique on 650 μm -thick SI-GaAs substrates. The structures consist of a $1 \times 10^{19} \text{ cm}^{-3}$ Be:doped 0.7 μm -thick bottom contact, 16 periods of 1250 Å-thick $\text{Al}_{0.12}\text{Ga}_{0.88}\text{As}$ undoped barrier/ 188 Å-thick Be:doped GaAs emitter followed by a $1 \times 10^{19} \text{ cm}^{-3}$ Be:doped 0.2 μm -thick top contact. The emitter layer doping density was 10^{10} , 3×10^{16} and $1 \times 10^{17} \text{ cm}^{-3}$ for detectors HE0204, HE0205 and HE0206, respectively. The detectors were fabricated by etching different size mesas using wet etching techniques. Ti/Pt/Au ohmic contacts were evaporated onto the top and bottom contact layers and a window was opened through the top contact for front-side illumination. The top contact was thinned to roughly 1000 Å (with ~ 100 Å depleted) leaving 900 Å of the top contact to serve as the first emitter layer.

4.1.2 Results and Discussion

Figure ??(a) shows the Dark current vs. Bias voltage for HE0204 at different device temperatures. These results are consistent with the dominant current mechanism being thermionic emission. The dotted line shows the noise level of the Keithly source-meter used in the measurements. A modified Arrhenius plot as shown in Fig. ??(b) gave an

activation energy of $\Delta \sim 60$ meV ($\lambda_0 \sim 20$ μm) at a bias voltage of 3.0 V. The activation energy increases as bias decreases, giving 70 meV ($\lambda_0 = 17.7$ μm) at 1.0 V. The photocurrent corresponding to the 300 K background (solid line with the label “Photo”) is also shown in Fig. ??(a). The photocurrent is obtained for a device temperature of 40 K. This shows a BLIP operation for bias < 1.5 V. As the temperature was decreased the bias range for BLIP operation increased, as shown in Fig. ??(c). For temperatures higher than 45 K, the dark current was always higher than the photocurrent and BLIP operation was not possible. A cold stop with FOV = 62° was used in the photocurrent measurements.

Figure 4.5(a) shows a comparison of dark current at $T = 77$ K for the three detectors with detector HE0206 showing a greatly decreased dark current. Samples HE0204 and HE0205 show only a small difference in dark current, which is probably related to changes in the bandgap narrowing from the doping. However, due to the same effective barrier height, one expects the same dark current for all three detectors. The variation between HE0206 and HE0205 is too large to be explained by bandgap narrowing effects alone which should be 1–2 meV at most. It is believed that for the lower doping density in HE0206, the impurity and valance bands in the emitter layer have not merged yet, leading to reduced thermionic current.^{47, 28} As the doping increases the upper and lower Hubbard bands increase in width until they merge. At this point, the impurity and valance bands merge to form a unified band, and the conduction should increase. The differences can be seen clearly at higher temperatures where the thermionic component is dominant. The dark current density at $T = 77$ K of devices with four different electrical areas are shown in Fig. 4.5(b). The high uniformity among the detectors is reflected by the fact that all four

dark current curves look as if the plot was for a single detector. This indicates that the edge leakage in HE0206 is negligible.

Figure 4.6(a) shows the measured responsivity in the range 5–20 μm for HE0204 for 4.2–50 K with a peak responsivity of ~ 0.1 A/W at 12.5 μm up to 40 K. The total quantum efficiency, determined by dividing the photocurrent by the incident photon rate, was ~ 0.8 %. The D^* value calculated from the dark current assuming full shot noise was $5 \times 10^{10} \text{ cm}\sqrt{\text{Hz}}/\text{W}$. The 30% points give a response range of 6–17 μm . The response was relatively stable up to around 40 K beyond which it decreased rapidly, disappearing by ~ 60 K. The responsivity of the detectors with lower doping density is greatly reduced, being 1.5 and 0.03 mA/W for HE0205 and HE0206, respectively at 12.5 μm . The device doping density and the maximum measured response are listed in Table 4.1. The response at $T = 4.2$ K for HE0205 is shown in Fig. 4.6(b). The responsivity of HE0206 is not shown as it was greatly reduced. Samples HE0205 and HE0206 showed some response up to temperatures of 50 and 45 K respectively, compared to the response at 55 K seen for detector HE0204. Although the decreased doping improved the dark current in HE0205, the responsivity was reduced drastically, reducing its D^* . This indicates that the use of high doping density may be the preferable approach. Based on both experimental results and the standard thermionic current calculation, the dark current will not increase significantly as doping is increased until the extremely high doping causes defects in the barrier. These defects will lead to an increase in the tunneling current of the detector. If the doping is kept below the very high values, the absorption is increased, so the response and hence the BLIP temperature should increase. It is possible to increase the doping density to at least

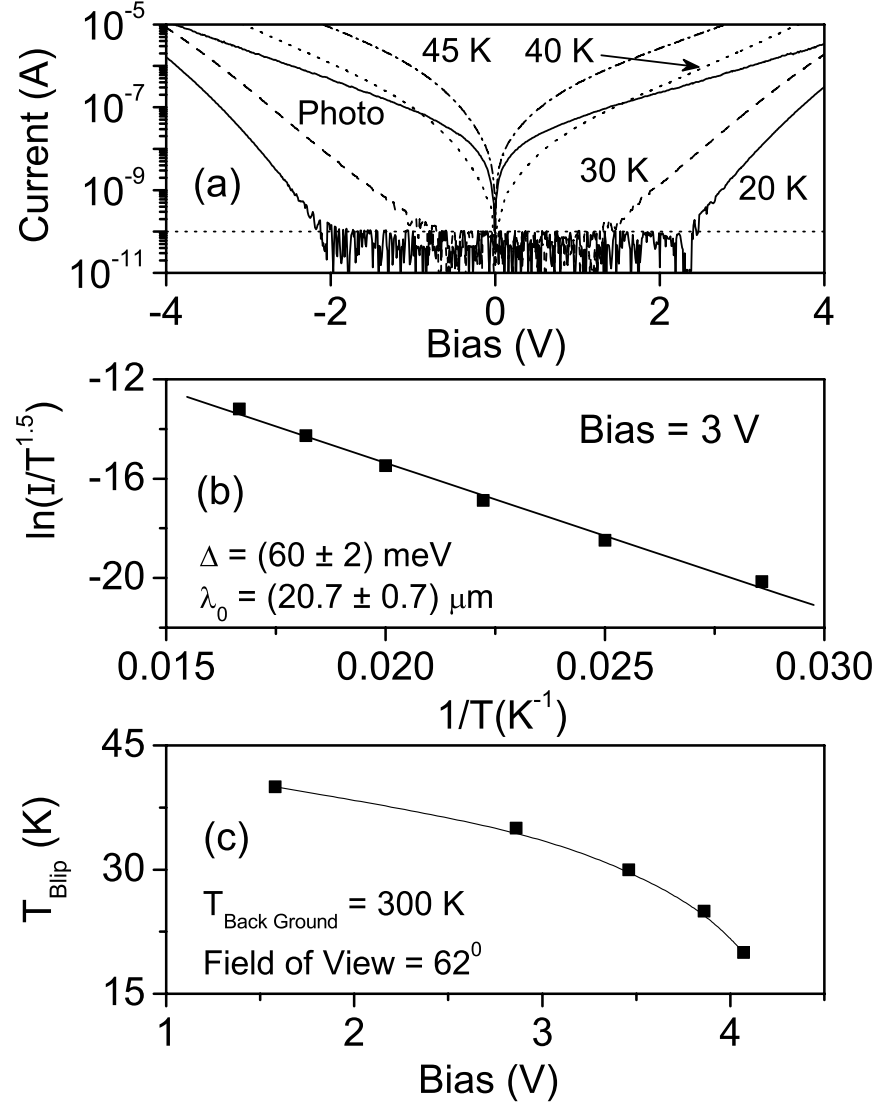


Figure 4.4: (a) Dark current vs. Bias at different device temperatures illustrating the thermionic nature of the current. Also shown is the 300 K background photocurrent obtained at 40 K (solid line). The temperature $T = 4.2 \text{ K}$ gives a maximum bias of 1.5 V for BLIP operation. (b) A modified Arrhenius plot giving a threshold wavelength of $\lambda_0 \simeq 20 \mu\text{m}$. (c) Plot of BLIP temperature vs. Maximum bias voltage. The region under the curve gives the combinations of bias voltage and device temperature that allow the detector to operate within BLIP condition.

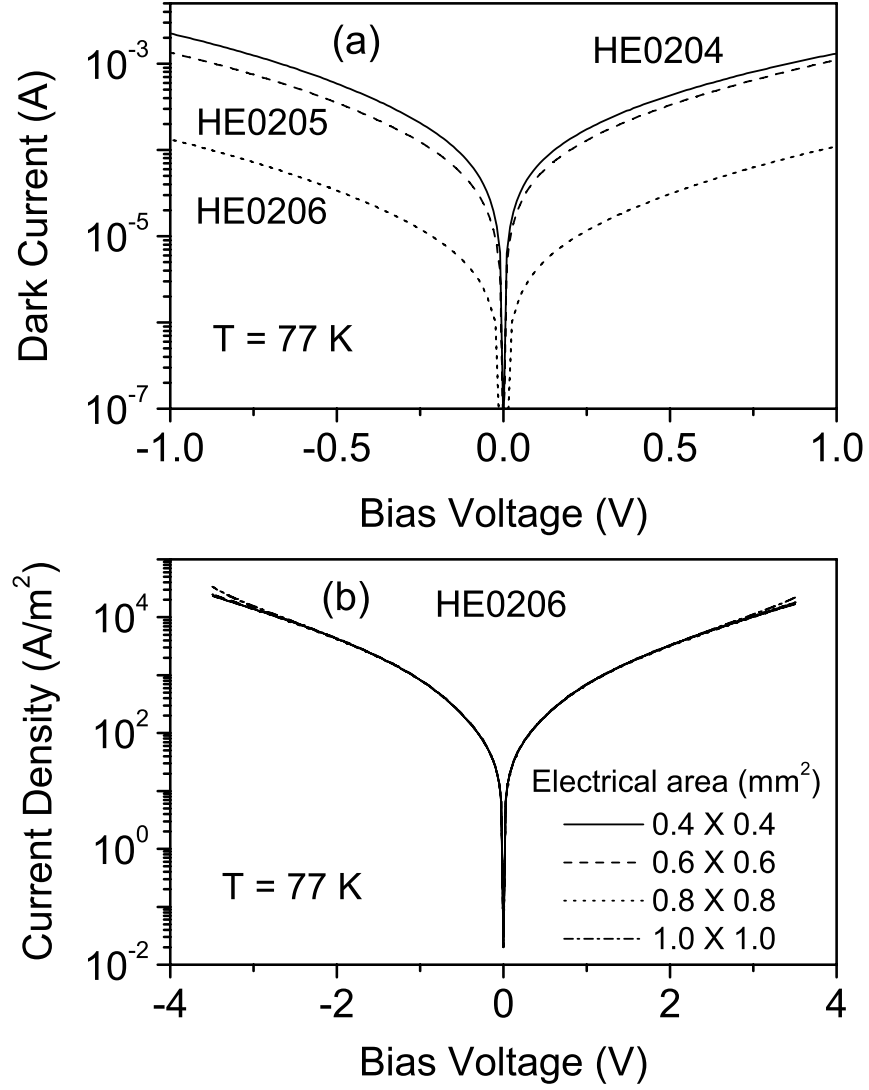


Figure 4.5: (a) Dark current at $T = 77$ K for the three detectors. The doping density in the emitters of HE0204, HE0205 and HE0206 are 1×10^{18} , 3×10^{17} and 1×10^{17} cm $^{-3}$, respectively. Similar variation is observed at $T = 4.2$ K where the current is much lower. (b) Comparison of dark current density between mesas of HE0206 with different electrical areas. The measurements were done at $T = 77$ K. The four curves appear essentially as a single curve due to high uniformity.

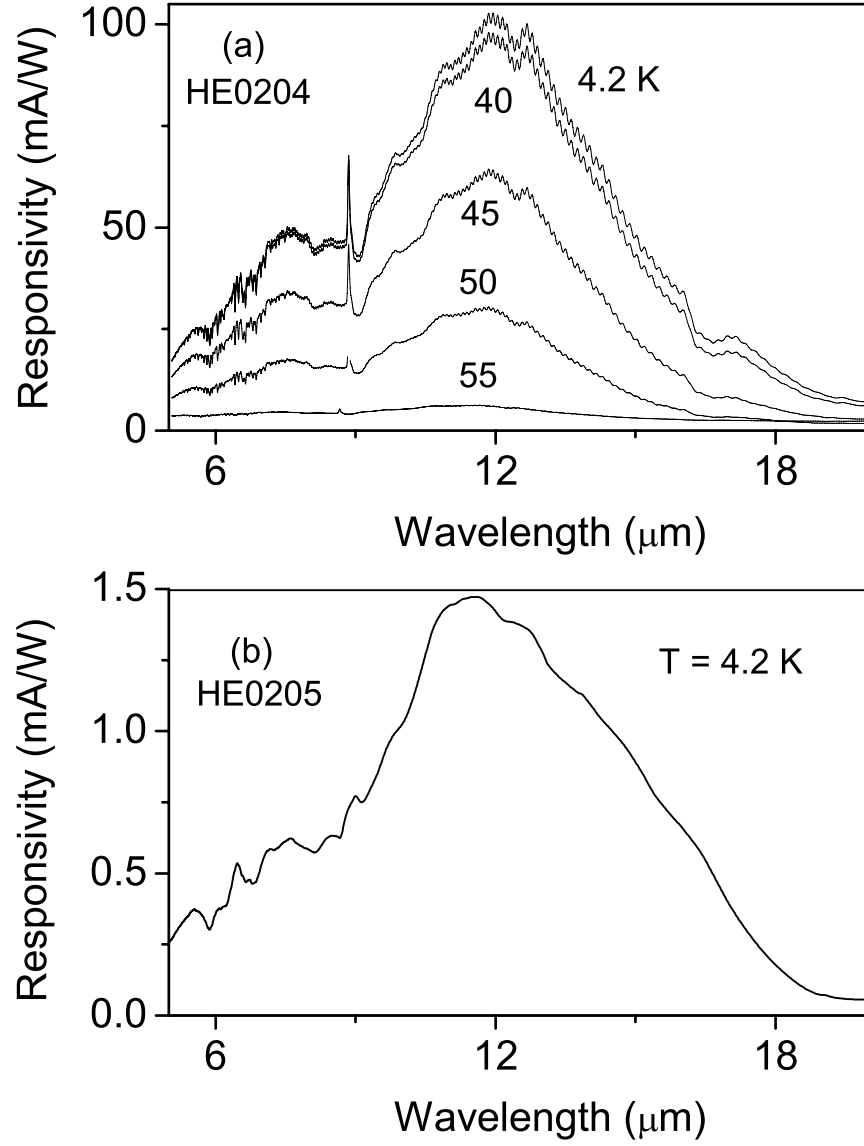


Figure 4.6: (a) Measured responsivity of HE0204 at different temperatures. The peak response was 100 mA/W at $\sim 12.5 \mu\text{m}$. The response remained nearly constant up to 40 K and then decreased rapidly. This is consistent with the BLIP temperature of 40 K estimated from dark and background currents. (b) Responsivity for HE0205 at 4 V bias and $T = 4.2$ K, showing the greatly reduced response observed when doping is reduced.

Table 4.1: Parameters of the detectors used in the measurements. In all cases, the emitters were 188 Å-thick GaAs, the barriers were 1250 Å-thick $\text{Al}_{0.12}\text{Ga}_{0.88}\text{As}$, and the contacts were doped to $1 \times 10^{19} \text{ cm}^{-3}$. For all three detectors, $\sim 900 \text{ Å}$ (assuming a depletion of $\sim 100 \text{ Å}$) of the top contact layer was left after etching to form the first emitter. Also shown are the measured values of responsivity and D^* .

Sample	Emitter doping $\times 10^{17}$ (cm^{-3})	Responsivity at 4.2 K (mA/W)	D^* ($\text{cm}\sqrt{\text{Hz}}/\text{W}$)	
			4.2 K	40 K
HE0204	10	100	2×10^{11}	5×10^9
HE0205	3	1.5	4×10^9	—
HE0206	1	0.03	4×10^8	—

$3 \times 10^{18} \text{ cm}^{-3}$, as will be discussed in the following paragraphs. To avoid increased tunneling from defects at even higher doping, inclusion of a thin undoped GaAs region between the barrier and emitter could be considered.

The drop in responsivity for HE0206 is much larger than expected. Based on theoretical calculations, the free carrier absorption should vary as N_A for phonon moderated processes and $N_A N_i$ for impurity moderated processes if the density of ionized impurities is N_i . Because the escape probability does not depend on the doping density, it is expected that the response should show the same doping dependence as the absorption. Then N_i is expected to be the same as the hole density N_A , giving a maximum dependence of N_A^2 for the responsivity for the fully ionized impurities. However, based on the experimental results the responsivity varies as $N_A^{3.5}$, which can be explained if the impurities are not fully ionized at low doping levels. This introduces an extra factor that accounts for the observed strong dependence of the responsivity on the doping in this regime.

The response can also be enhanced by using resonant cavity effect³⁰. By designing the device for improved reflection from the bottom contact, a resonant cavity can be employed with associated enhancement of the absorption at specific wavelengths. One way to improve reflection is through the use of a *n*-type rather than a *p*-type bottom contact due to the differences in their skin depths and refractive indices. Figure 4.7(a) shows the calculated absorption in the top emitter for HEIWIP structures with a $1 \times 10^{19} \text{ cm}^{-3}$ 200 Å-thick *p*-GaAs top contact (serving as the top emitter), 4 periods of $1 \times 10^{18} \text{ cm}^{-3}$ 188 Å-thick *p*-GaAs emitters and 1250 Å-thick undoped $\text{Al}_{0.12}\text{Ga}_{0.88}\text{As}$ barriers, $1 \times 10^{19} \text{ cm}^{-3}$ 5000 Å-thick *p*-GaAs bottom contact layer followed by a SI GaAs substrate. These are designed to

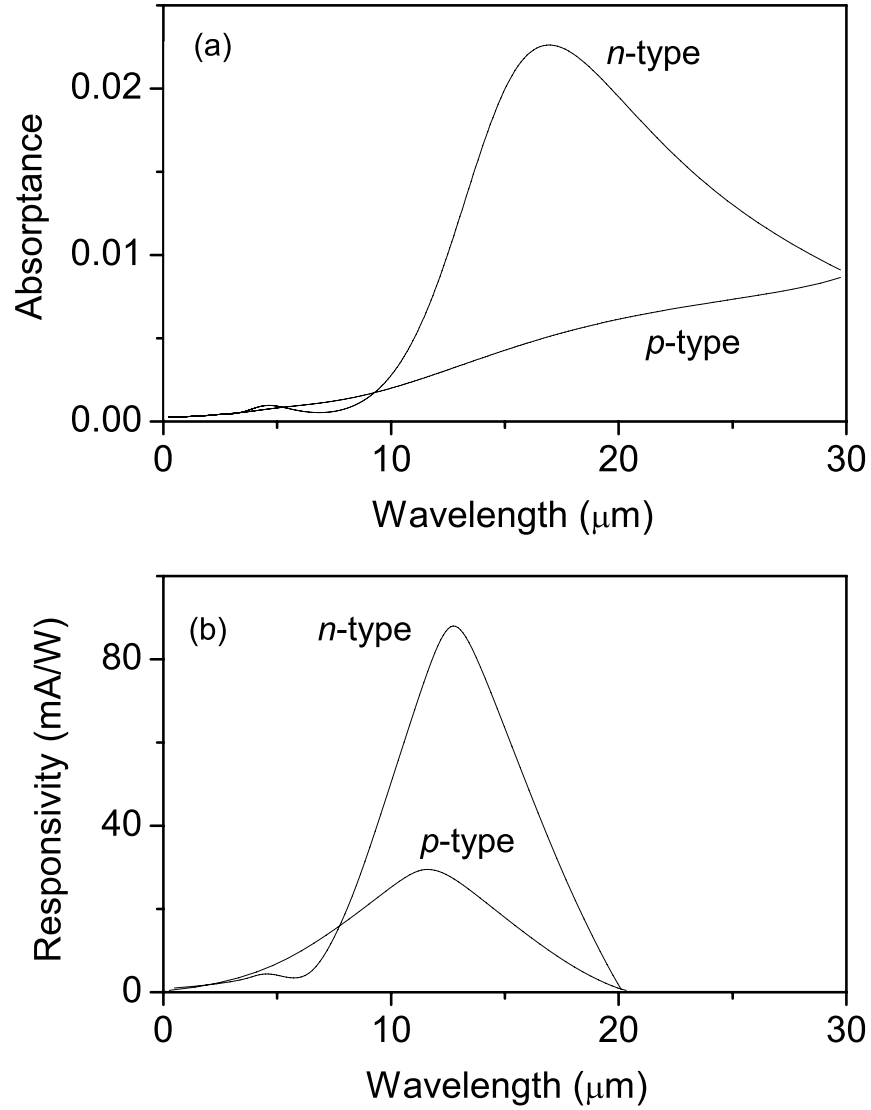


Figure 4.7: (a) Model absorptance in the first emitter (200 Å-thick remainder of the top contact) for devices with *n*- and *p*-type bottom contacts. Both designed with the same parameters as of HE0204, but with reduced emitter/barrier units to have the first order resonance peak near 15 μm . The use of *n*-type material greatly increases the absorption in the 10–20 μm range. (b) Model response for the two designs. The use of *n*-type material leads to almost a factor of 4 increase in the peak response.

be similar to the detectors used in the measurements, but with the doping density increased and the number of emitter/barrier units reduced so as to obtain the first order resonance peak at $\sim 15 \mu\text{m}$. A comparison was made for n - and p -type doping in the bottom contact. As can be seen, the use of n -type doping for the bottom contact will give a much higher absorption at shorter wavelengths than for a p -type contact. In addition, the increased skin depth³⁰ for the p -type material leads to optimum absorption at longer wavelengths than are desired for detectors operating below $20 \mu\text{m}$. As a result, the n -type bottom contact serves better in reflecting radiation of wavelength below $20 \mu\text{m}$. Increasing the bottom contact thickness will also increase the reflection for both types of contact, and a p -type contact has the effect of shifting the peak absorption to shorter wavelengths. However, there are practical limits on the thickness of contact layers that can be grown. To avoid these limits a doped substrate can be used. Then the substrate can also serve as the contact layer, avoiding the need to include a separate bottom contact in the design.

The model responses for the structures with n - and p -type bottom contact layers are shown in Fig. 4.7(b). The peak response of the design with p -type bottom contact, $\sim 20 \text{ mA/W}$, is based on just the top emitter. Therefore, such a low value can be justified compared to the measured peak response, $\sim 100 \text{ mA/W}$, of HE0204. Clearly, the difference is the result of excluding other emitters in the calculation. For a n -type bottom contact, based on the model curves in Fig. 4.7(b), the response would be expected to increase by a factor of 4. This means that if the bottom contact layer of HE0204 was n -type, then the measured peak response would have been $\sim 400 \text{ mA/W}$. For such a device, assuming the dark current is not increased by the doping, so the noise will be similar to HE0204, the pre-

dicted BLIP temperature could be ~ 55 K with a corresponding D^* of $2 \times 10^{10} \text{ cm}\sqrt{\text{Hz}}/\text{W}$. By using higher doping (e.g., $3 \times 10^{18} \text{ cm}^{-3}$) the device responsivity can be further increased, giving even higher BLIP temperatures.

4.1.3 Conclusion

In conclusion, HEIWIP detectors operating in the 8–20 μm range with a maximum responsivity of 0.1 A/W and $D^* = 2 \times 10^{11} \text{ cm}\sqrt{\text{Hz}}/\text{W}$ at $\sim 12.5 \mu\text{m}$ and 4.2 K were demonstrated. At $T = 40$ K, the peak responsivity was 95 mA/W with D^* of $5 \times 10^9 \text{ cm}\sqrt{\text{Hz}}/\text{W}$. The BLIP temperature of the detectors was ~ 40 K and response was observed up to ~ 60 K. Although the dark current decreased with decreasing emitter doping, the response decreased faster, leading to the device with the highest doping having the best response. With structures optimized for this wavelength range, detectors with strong response should be obtainable.

4.2 Effect of Doped Substrate on Resonant Cavity Enhancement

The progress in epitaxial growth technology facilitates resonant cavity architecture designs in optoelectronic devices. The RCA has been used to enhance response in many devices¹⁷ including QWIPs⁴⁸, photodiodes, phototransistors, modulators, and LEDs. Placing an optoelectronic device or the active area of it in a resonance cavity increases the quantum efficiency of these detectors. For HEIWIP detectors, the implementation of RCA is a two step process, where the first step is to use the reflection from the bottom layer to provide standing waves in the device. The intensity of such standing waves is controlled by the reflectance of the bottom layer. The second step is to place the emitters at the anti-nodes of the standing waves that correspond to the wavelength desired for enhancement. The effect of RCA has already been seen even in structures with SI-substrates³⁷. The use of doped substrates enhances the quantum efficiency of a detector, and the spectral region of interest for such enhancement is determined by the cavity thickness of the detector. This section discusses the role of doped substrates as reflectors in enhancing the cavity resonance in HEIWIP detectors.

The substrates used in IWIP detectors are typically 400–600 μm -thick undoped bulk layers. Therefore, the reflection from these substrates cannot be used to form low order resonance in the 1–50 μm response range, so strong resonance from the epitaxial cavity cannot be expected. Two possible approaches to developing a usable resonant cavity architecture in devices are: i) to use a doped substrate as the reflector or, ii) to grow a reflecting structure below the active region of the device. The reflecting structure could

be as simple as a single doped layer or could involve multiple doped and undoped layers as in a Bragg reflector. In some cases, the bottom contact could also be the reflector, or the reflector may be grown separately. Here, results of a detector tested based on the first approach, i.e., a detector structure grown on a n -type substrate, are discussed.

The effectiveness of a reflecting layer depends on its thickness and the skin depth δ . For layers with thickness less than 3δ , a significant fraction of the radiation will be transmitted rather than reflected. The results of skin depth calculated from the complex dielectric constant derived from Eq. 2.19 for both n - and p -type materials are shown in Fig 2.12. For longer wavelengths ($\lambda > 40 \mu\text{m}$), the skin depth stays relatively constant, with values between $\sim 40 \mu\text{m}$ for p -type material doped to $1 \times 10^{18} \text{ cm}^{-3}$ and $0.6 \mu\text{m}$ for n -type material doped to $1 \times 10^{19} \text{ cm}^{-3}$. The skin depth increases as the wavelength decreases, becoming larger than $100 \mu\text{m}$ at wavelengths shorter than $10 \mu\text{m}$ for p -type doping. The lower effective mass for electrons compared to holes leads to the reduced skin depth in n -type material. Although reflection is improved by the use of n -type layers, a device based totally on n -type material is not optimum for IR detection. For lower carrier mass, the absorption coefficient will increase; however, the increase in Fermi level will greatly reduce the photoemission probability. The relative magnitude of these two effects will depend on the doping in the emitters. For low doping levels, the lighter carriers will be more efficient, and at higher doping heavier carriers become more efficient. At the doping levels in the devices discussed ($\sim 3 \times 10^{17} \text{ cm}^{-3}$), the efficiency of a n -type emitter material is about 0.95 times the efficiency for a p -type. Experimental results indicate that higher doping densities than those used here are desirable; hence, the optimum design should use a n -type reflecting

layer as the bottom contact with p -type emitters to enhance its photoresponse.

For a typical HEIWIP device, p -type $\sim 10^{19} \text{ cm}^{-3}$, $0.7 \text{ }\mu\text{m}$ -thick bottom contacts are grown⁴⁹. This is less than the skin depth at all wavelengths. Hence a significant fraction of the incident radiation will pass through the bottom contact rather than being reflected, reducing the strength of the resonance amplitude. Even for a highly n -type material, the thickness is only similar to the skin depth at long wavelengths ($> 10 \text{ }\mu\text{m}$ for $1 \times 10^{19} \text{ cm}^{-3}$ n -type material). By increasing the reflecting layer thickness to $\sim 1.5 \text{ }\mu\text{m}$ for n -type material an effective cavity could be formed at $\lambda > 10 \text{ }\mu\text{m}$. However, for shorter wavelengths, growing a thick enough reflecting layer will not be feasible due to the larger skin depth. The skin depth limitation is responsible for the difficulty in clearly identifying resonance maxima in previous p -type HEIWIP detectors⁵⁰. One solution to the difficulty is to use a doped substrate to function as the reflecting layer as well. A doped substrate should increase the reflection from it near the active region without the need to grow a thick reflecting layer.

4.2.1 Experimental

Two GaAs/AlGaAs detectors, 1329 with a doped substrate and HE0205 with an intrinsic substrate, were used to demonstrate the resonant cavity enhancement from the doped substrate. The structure parameters for the two detectors are given in Table 4.2. The significant difference between these was the use of a SI-substrate in HE0205 and a n -type substrate in 1329. Both devices used GaAs emitters and $\text{Al}_{0.15}\text{Ga}_{0.85}\text{As}$ barriers. However, HE0205 had p -type emitters whereas 1329 had n -type emitters. From calculations using Eq. 2.30, it was expected that n -type emitters would be 95% as efficient as p -type emitters (see Fig. 2.14 for comparison). Hence, even the small difference expected for the n -type

Table 4.2: Device parameters of two detectors (labeled HE0205 and 1329) used to demonstrate the resonance cavity effect. These detectors have the same parameters, except for HE0205 being p -type and grown on a SI-substrate whereas 1329 is n -type and grown on a n -type substrate, with emitters doped to $\sim 5 \times 10^{18} \text{ cm}^{-3}$. Both detectors have 12 emitter/barrier units and the thickness of emitters and barriers are 188 Å and 1250 Å, respectively.

Sample	Emitter doping $\times 10^{17} \text{ (cm}^{-3}\text{)}$	Contact doping $\times 10^{18} \text{ (cm}^{-3}\text{)}$	Peak Response (mA/W)	Substrate type
HE0205	3 (p)	10	1.5	SI
1329	4 (n)	1	12	n

layers will reduce the response. Therefore, the increase observed in the response should be due to cavity effects.

4.2.2 Results and Discussion

The dark current for both devices were measured at different temperatures, and Δ vs Bias variations were obtained using Arrhenius plots. A significant feature was a rapid decrease in the barrier height Δ observed for 1329 under forward bias and a slower decrease under reverse bias as shown in Fig. 4.8. Typically, Δ is expected to vary about 10 meV at these doping levels. However, for 1329 it decreases from 95 to 45 meV as the bias is varied from 0 to 0.7 V.

This large variation is believed to be due to the difference in doping in the contact and emitter layers, and could be used to tune λ_0 by varying the bias. The doping in the contact layer is $2 \times 10^{18} \text{ cm}^{-3}$, giving a much higher Fermi level than in the emitters which were doped to $4 \times 10^{17} \text{ cm}^{-3}$. The difference in these Fermi levels will lead to band bending in the first barrier layer. Consequently, the conduction band edge under zero bias would slope up from the contact towards the emitter as seen in the top inset of Fig. 4.8. The effective barrier height is determined by the difference between the Fermi level and the maximum height of the barrier. When a bias is applied to the device, the effective barrier height can be determined from an Arrhenius plot of $\ln(I/T^{3/2})$ vs. $1/T$. Under a small bias (middle inset in Fig. 4.8), the highest energy for the conduction band will still be adjacent to the emitter. However, it will be reduced due to the applied field in the barrier causing the barrier to be bias dependent. However, above a certain bias, the drop due to the applied field will exceed the effects of the band bending and the barrier will slope down from the

contact to the emitter (bottom inset in Fig. 4.8).

The effective barrier is now determined by the barrier height adjacent to the contact which will then vary only slowly with bias. The difference in behavior for forward and reverse bias is probably related to small differences between the doping in the two contacts. The fact that the variation is seen only in the forward and not in the reverse direction means that a large difference in λ_0 will be seen between the forward and reverse directions. Because of this λ_0 variation, comparison of response in the forward and reverse directions will have to be limited (to less than the peak wavelength) to obtain valid results. When the resonant cavity effect is ignored⁴⁹, the response increases linearly with the wavelength for short wavelengths, reaching a maximum, and then decreases to zero at λ_0 . Previous results⁴⁹ (both model and experimental) for detectors with different λ_0 have shown that for the short wavelength region where response increases linearly, the response does not depend on λ_0 . However, for longer wavelengths, response increases as λ_0 is increased. By restricting the comparison to the linear region ($< 10 \mu\text{m}$ for detector 1329), the difference in λ_0 between forward and reverse bias can be ignored.

The contacts can also serve as emitters for the devices and can make a significant contribution to the device response. For HE0205, the contact doping is $1 \times 10^{19} \text{ cm}^{-3}$, and the emitters are only doped to $3 \times 10^{17} \text{ cm}^{-3}$. Based on the device model, response from the top contact should be almost 10 times the response from the emitters and 30 times the response from the bottom contact. The forward bias response for HE0205 with a peak response of 1.5 mA/W is shown in Fig. 4.9, and the reverse bias response was not measurable. The two key properties of this detector are the use of a SI-substrate and p -type

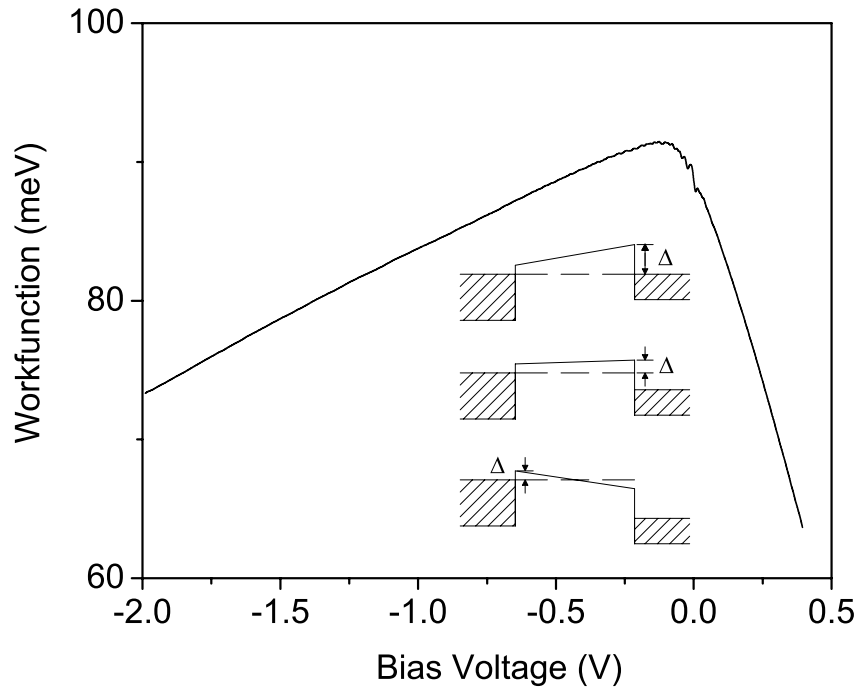


Figure 4.8: Variation in the experimental workfunction with the bias for detector 1329 with a doped substrate. The workfunction drops rapidly in the forward bias, and varies slowly for reverse bias. The inset shows a band diagram of the first barrier with the contact on the left and an emitter on the right. The higher doping in the contact gives a higher Fermi level. Under zero bias (the top diagram), the barrier slopes up with the highest value adjacent to the emitter. Under low bias (the middle diagram), the barrier slopes up with the effective barrier height Δ reduced by the applied bias. Under high bias (the bottom inset), the barrier slopes down. Here, Δ is determined from the contact and has only a slow variation with bias.

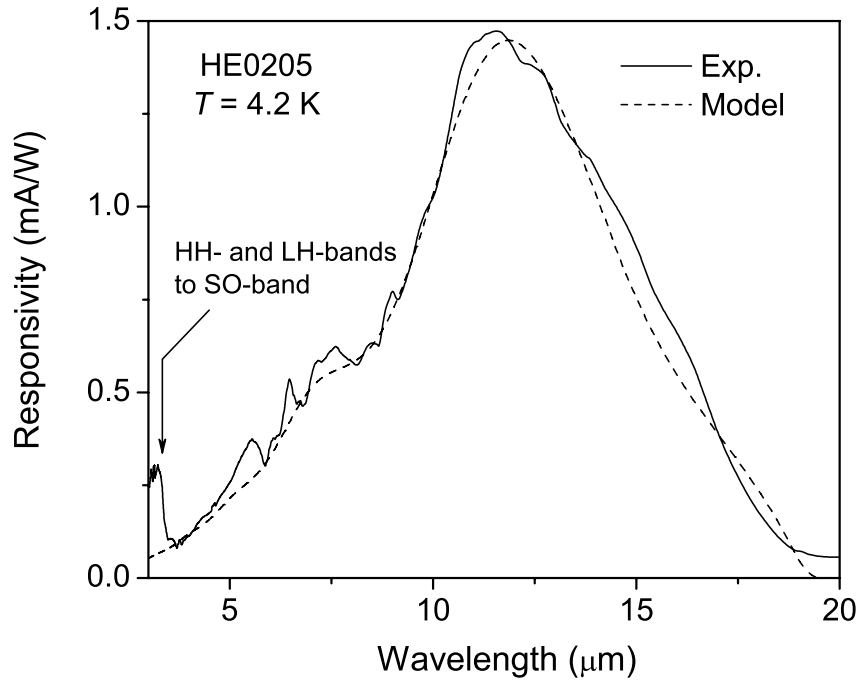


Figure 4.9: Response under forward bias for detector HE0205 at $T = 4.2$ K. Response was not observed under reverse bias. The dashed line shows the modeled response for the detector parameters of HE0205. The arrow at $\lambda \simeq 3 \mu\text{m}$ indicates the response due to carriers photoexcited from heavy-hole (HH) and light-hole (LH) bands to split-off (SO) band of GaAs. This mechanism is not included in the model.

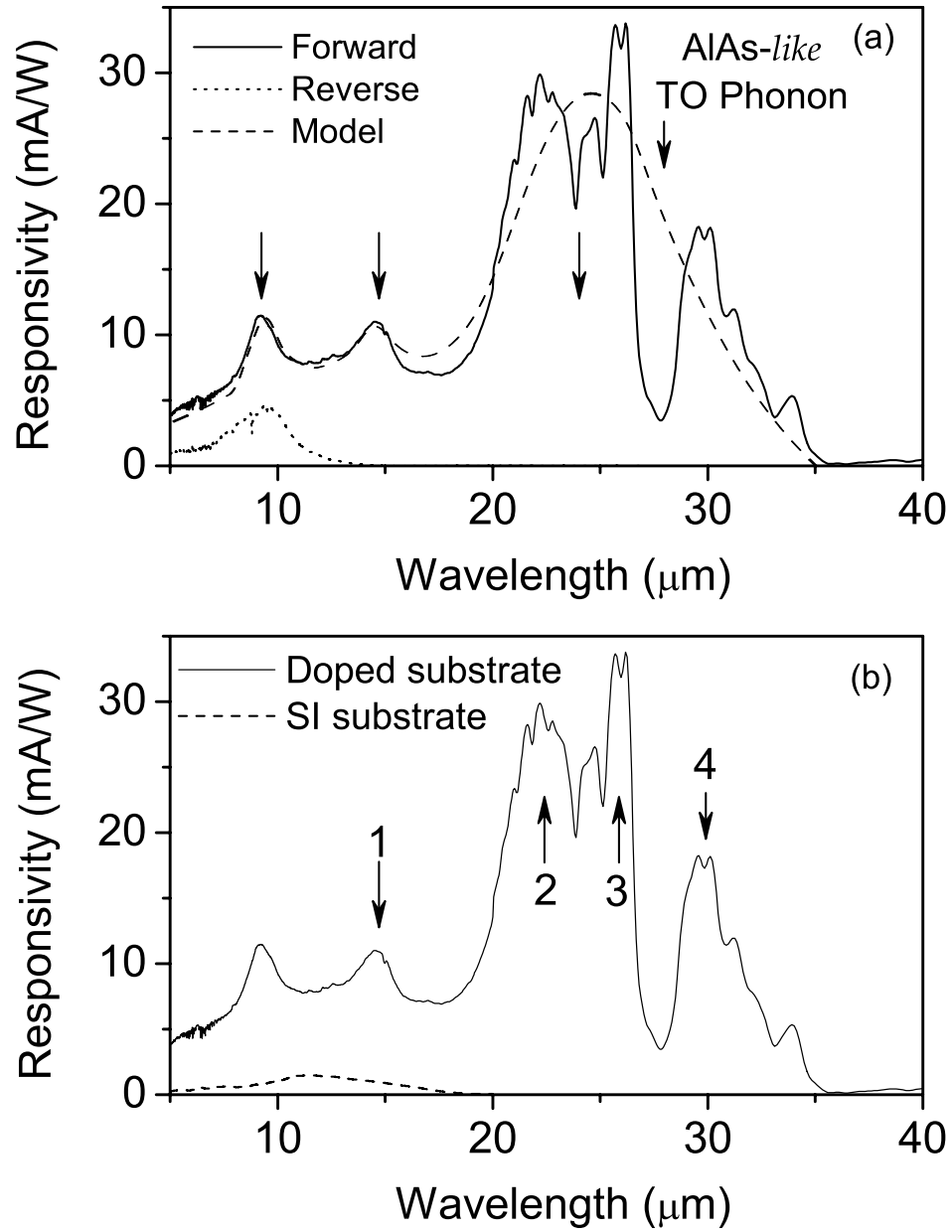


Figure 4.10: (a) Response under forward and reverse bias for detector 1329 (grown on a n -type substrate) at $T = 4.2$ K. The response is strong under forward bias. The peak at $10 \mu\text{m}$ is enhanced by waveguide effects under the contact region. The dashed line shows the modeled response for the same detector parameters. (b) Comparison of response for structures grown on doped and SI-GaAs substrates. As expected, the numbered peaks correspond to minima of the reflectance spectra shown in Fig. 4.11.

emitters. Any reflection in the undoped substrate structure must come from the bottom contact. This means that the optical electric field, and hence the absorption, should be small in the bottom contact, compared to the top contact as was observed experimentally. For a p -type device, forward bias causes the top contact to serve as an emitter with a strong response and reverse bias causes the bottom contact to serve as an emitter with weak or no response.

The response under forward and reverse bias for detector 1329 is shown in Fig. 4.10(a). Response was seen for both cases, although the forward bias signal of 12 mA/W at 9 μm was a factor of 4 stronger than the reverse bias signal of 3 mA/W. The comparison was done at 9 μm so that the difference in λ_0 for forward and reverse bias should not affect the responsivity. Again, the observed characteristics of the response are consistent with the resonant cavity enhancement. Sample 1329 was grown on a n -type substrate for which the reflection will occur inside the substrate at a point near the skin depth from the substrate/bottom-contact interface.

This means that absorption will occur in both the top and bottom contacts. Under forward bias, three cavity peaks are seen corresponding to the first, third, and fifth orders. Because the skin depth varies with wavelength, the effective thickness of the cavity will also vary with wavelength, and the observed peaks do not necessarily have frequencies that are multiples of the fundamental. When resonance peaks are fitted using a basic model for the HEIWIP response⁴⁹, with the absorption calculated from the resonant cavity effect as was done previously³⁰ and a gain of 2, a good agreement with the experimental data was obtained as shown in Figs. 4.9 and 4.10(a). The fit for the reverse bias response for

detector 1329 matches the experimental curve but is not shown for simplicity. To obtain the fit, the workfunction for the contact was assumed to be the value obtained from the Arrhenius plots whereas the workfunction for the emitters were taken as 90 meV according to SIMS data. The difference in workfunction introduces a threshold for the emitters just longer than the fifth order resonance peak, leading to its enhancement.

A response comparison of these detectors under forward bias indicates the presence of a resonance cavity enhancement in detector 1329, which was 8 times the response for HE0205 detector with the SI-substrate, as shown in Fig. 4.10(b). This is due to the efficient cavity formation caused by the reflection from the doped substrate. The increased resonant cavity nature of 1329 can also be seen from the comparison of its reflectance with that of HE0205 as shown in Figs. 4.11(a) and 4.11(b). For detector 1329, the average reflection is much higher than for the detector with an undoped substrate (HE0205). The reduced reflection strengths at the resonance wavelengths (reflection minima indicated by 1–4 in Fig. 4.11(b)) reflects in the response peaks indicated by the arrows labeled 1–4 in Fig. 4.10(b). The unlabeled response maximum at $10\text{ }\mu\text{m}$ does not appear in the absorption as it is believed to be related to the effect of metalization on the detector, which is not present on the structure pieces used for absorption measurements. Light that reflects under the metal contact undergoes resonance at both the top and bottom of the device for wavelength $\sim 10\text{ }\mu\text{m}$. The differences predicted by the model for p - and n -type emitters with a SI-substrate and for n -type emitters with a n -type substrate are shown in Fig. 4.12. The change in emitter type has only a minimal effect on the response whereas the n -type substrate produces a larger increase.

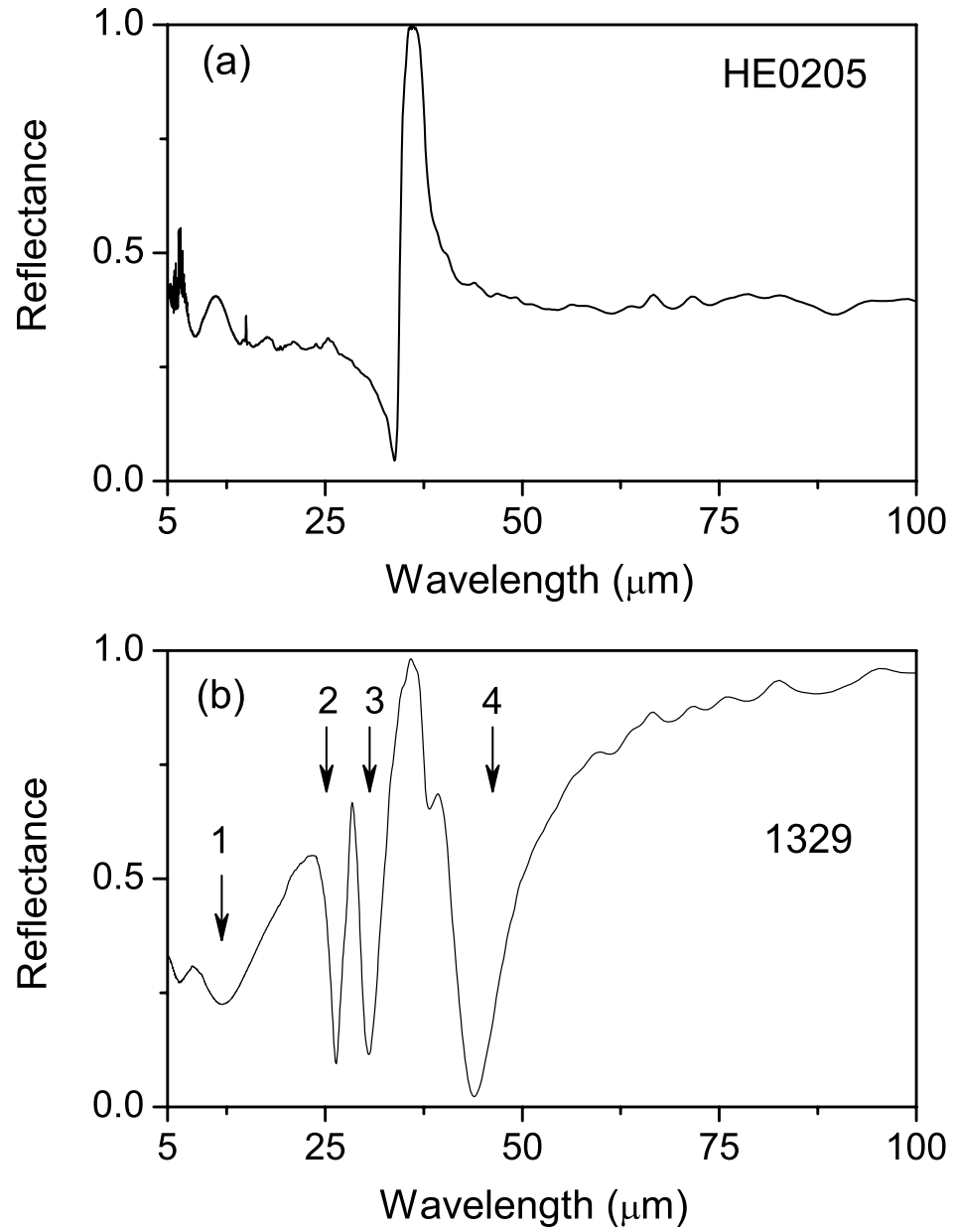


Figure 4.11: Reflectance measurements for detector structures (a) HE0205 and (b) 1329. The measurements show an increased reflection from detector structure 1329 as well as the reduced reflection (labeled 1–4) corresponding to the response peaks labeled in Fig. 4.10.

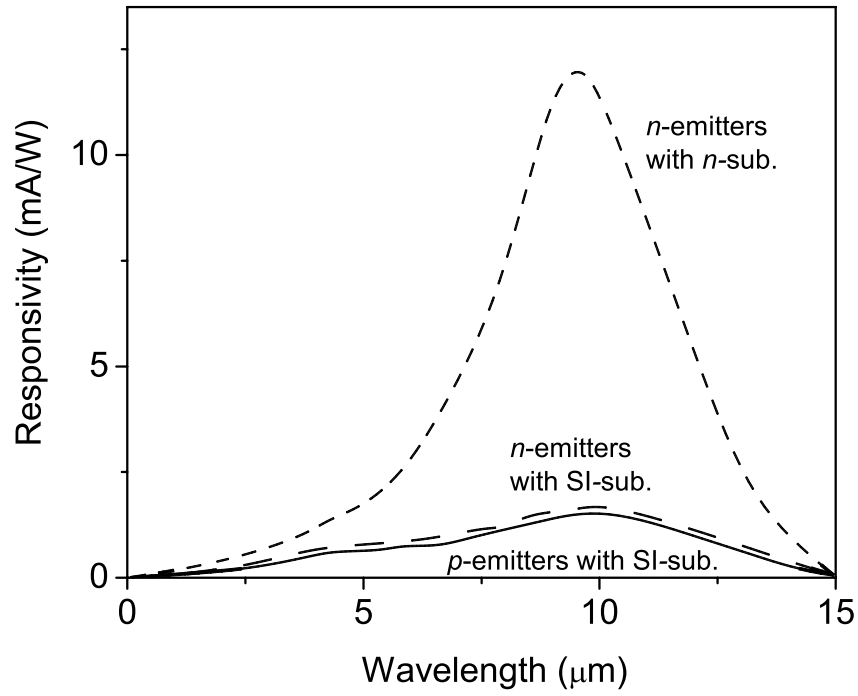


Figure 4.12: The predicted difference in response for devices with n -type emitters grown on a n -substrate, n -type emitters grown on a SI-substrate, and p -type emitters grown on a SI-substrate. Other parameters are the same as for HE0205. The change from p - to n -type emitters produces only a minimal change whereas the use of a n -type substrate produces an increase of ~ 8 times in the detector response.

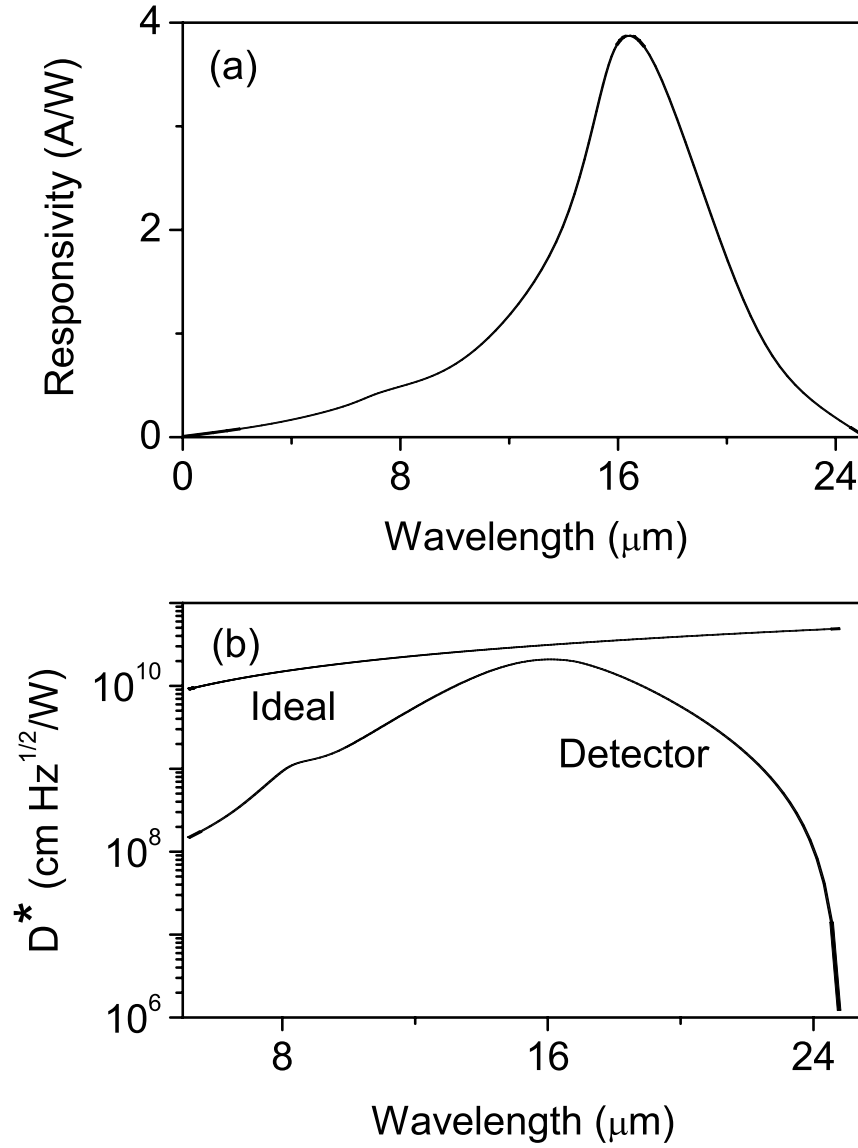


Figure 4.13: (a) Response for an optimized device with 32 periods of 200 Å-thick *p*-type GaAs emitters doped to $3 \times 10^{18} \text{ cm}^{-3}$ and 350 Å-thick undoped $\text{Al}_{0.15}\text{Ga}_{0.85}\text{As}$ barriers grown on a *n*-type GaAs substrate of doping density $5 \times 10^{18} \text{ cm}^{-3}$. The peak responsivity has been increased to near 4 A/W. (b) The calculated D^* with a peak value of $\sim 2 \times 10^{10} \text{ cm}\sqrt{\text{Hz}}/\text{W}$ as well as the calculated D^* for an ideal detector.

4.2.3 Conclusion

These results can be used to optimize a detector operating in the 12–20 μm range. The doping density in the emitters of the measured devices was only $3 \times 10^{17} \text{ cm}^{-3}$. By increasing the emitter doping density to $3 \times 10^{18} \text{ cm}^{-3}$, so that response from the emitters is similar to that from the contacts, a device with improved response can be obtained. A n -type substrate would be used to provide the maximum resonant cavity enhancement. Hence, a HEIWIP detector with 32 units of 200 Å-thick p -type GaAs doped to $3 \times 10^{18} \text{ cm}^{-3}$ and 350 Å-thick undoped $\text{Al}_{0.15}\text{Ga}_{0.85}\text{As}$ barriers grown on a n -type GaAs substrate should be near the optimum for operation in the range 12–20 μm with $\lambda_0 \sim 24 \mu\text{m}$. The responsivity of such a device as shown in Fig. 4.13 should be near 4 A/W with a quantum efficiency of 50% and $D^* \sim 2 \times 10^{10} \text{ cm}\sqrt{\text{Hz}}/\text{W}$ at 77 K corresponding to a BLIP temperature of 50 K.

4.3 Threshold Tailorability in Heterojunction Terahertz Detectors

The threshold frequency of a HEIWIP detector can be determined directly from $f_0 = \Delta/4.133$ where f_0 is in THz and the workfunction Δ is in meV. The Al fraction x can be reduced to any value in theory. However a practical lower limit will be around $x \geq 0.005$ with $f_0 \geq 2.7$ THz. (Although $x = 0$ is practical, then the device will no longer be a HEIWIP, and will have a ~ 9 meV offset for $N_A = 3 \times 10^{18} \text{ cm}^{-3}$ giving $f_0 = 2.17$ THz). Further decrease in f_0 below 2.7 THz (i.e. beyond $\sim 110 \text{ }\mu\text{m}$) will require a change in the design due to the minimum $\Delta = \Delta_d$ from the bandgap narrowing. One possible approach to avoid this limit is to use AlGaAs as the emitter and GaAs as the barrier. In such a device, the bandgap narrowing in the doped AlGaAs will be partially offset by the increased bandgap of the AlGaAs material relative to the GaAs, as seen in Fig. 4.14, giving $\Delta = \Delta_d - \Delta_{Al}$. For example, a $f_0 = 0.9$ THz ($335 \text{ }\mu\text{m}$) detector would have an Al fraction of ~ 0.01 . Based on calculations the FIR absorption in AlGaAs is expected to be very similar to that in GaAs, due to the very low Al content giving performances similar to the current devices with AlGaAs barriers.

The use of a heterojunction in HEIWIP detectors has several advantages over the homojunction approach. First is the ability to vary the threshold frequency (f_0) or wavelength (λ_0) while keeping the doping density (N_A) constant. This allows the doping to be adjusted to obtain the optimum combination of dark current and responsivity while not changing λ_0 . Second, the effective interface in a heterojunction is sharper than in HIWIP detectors due to the absence of space charge effects⁵¹. Third, for high doping, effects of

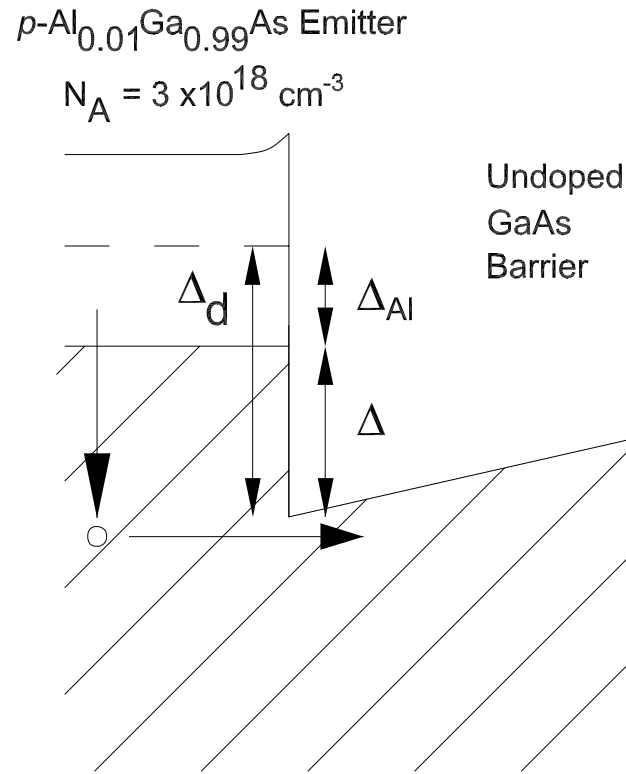


Figure 4.14: Band diagram of the emitter/barrier interface for a device using doped AlGaAs as the emitter and GaAs as the barrier to extend f_0 beyond 2.7 THz. The parameters shown are for a device with $f_0 = 0.9$ THz ($\lambda_0 \sim 335 \mu\text{m}$). The dashed line in the emitter indicates the Fermi level location if the emitter was GaAs. The contributions to Δ from the doping (Δ_d) and Al fraction (Δ_{Al}) are indicated by the vertical arrows. The effective barrier is $\Delta = \Delta_{Al} - \Delta_d$.

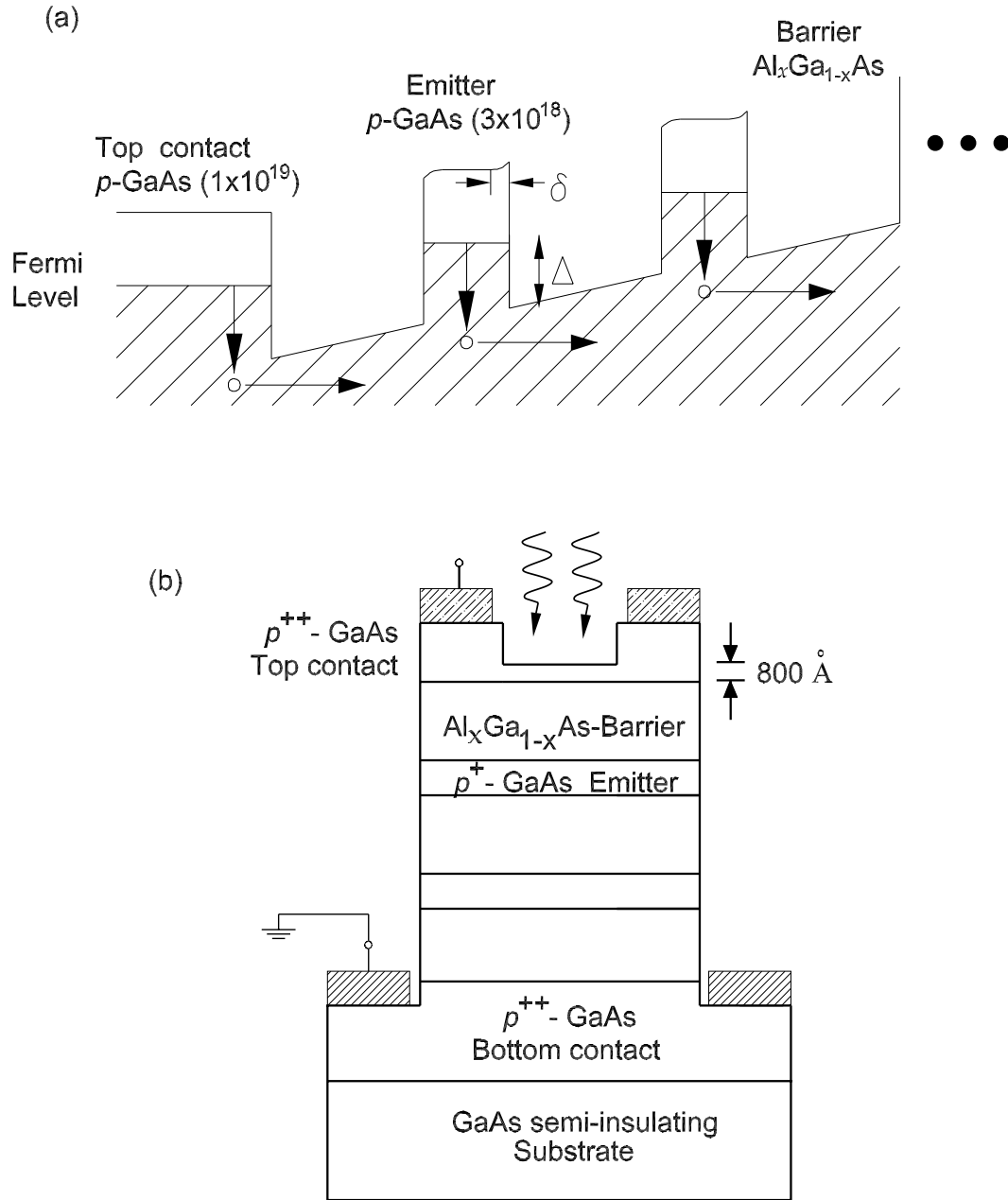


Figure 4.15: (a) A partial band diagram of the top two periods of a HEIWIP detector with doped GaAs as the emitter layer and undoped AlGaAs as the barrier. The effective width of the nonzero field region is indicated by δ . The emitter doping forms a 3-D carrier distribution. The detectors have $3 \times 10^{18} \text{ cm}^{-3}$ Be-doped 158 Å-thick emitters and 800 Å-thick $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barriers. (b) Schematic diagram of the detectors after processing. A window is opened on the top for front illumination.

heavy-light hole transitions can limit λ_0 for HIWIP detectors. Because the doping does not need to be as high in HEIWIP detectors at long λ_0 , this limit can be avoided.

This section discusses the design of HEIWIP detectors and also presents experimental results showing variation of f_0 (λ_0) with Al fraction. Two important performance factors for the design of a HEIWIP detector are the spectral response and dark current. The spectral response is characterized by f_0 or λ_0 where the response goes to zero, the peak responsivity (f_p or λ_p) where the response reaches its peak, and the peak quantum efficiency η_p . The overall goal is to maximize response while minimizing the dark current to obtain a high D^* . The basic approach to calculating the responsivity for a HEIWIP detector is somewhat similar to that done previously for the HIWIP detector²⁸. The difference being the origin of the barrier height workfunction and the effect of space charge. The responsivity is given by:

$$R = \frac{e\eta}{hf}, \quad (4.1)$$

where η is the total quantum efficiency.

The total quantum efficiency is the product of the photon absorption efficiency and the internal quantum efficiency (the probability that photoexcited carriers undergo internal photoemission) $\eta = \eta_a \eta_i$. Here, the collection efficiency is assumed to be 1 as the maximum barrier height is at the interface due to the absence of space charge effects, so that any carriers scattered after internal photoemission will be collected.

The photon absorption efficiency of a single layer is $\eta_a = (F/F_0)^2(1 - \exp(-\alpha W))$ where α is the wavelength dependent absorption coefficient, W is the emitter layer thickness, F is the optical electric field in the emitter layer, and F_0 is the optical electric field incident

on the device. The factor $(F/F_0)^2$ involving the electric fields adjusts for any resonance effects in the device structure, reflection at the top surface, etc. For high frequency operation resonant cavity effects can be significant, whereas at lower frequencies they are generally less important. For frequencies greater than ~ 38 THz ($8 \mu\text{m}$), the absorption is a combination of free carrier and intersubband absorption⁵² that varies as $1/f^2$. The absorption is nearly independent of frequency below 38 THz ($> 8 \mu\text{m}$). The internal quantum efficiency can be calculated using the same model used for HIWIP detectors²⁸ with Δ replaced by the HEIWIP value. The internal quantum efficiency η_i is zero at the threshold frequency f_0 and increases continually as the frequency is increased. The shape of η_i varies with doping density shifting the response towards longer wavelengths as the doping is decreased with Δ held fixed by varying x .

It is thus seen that the primary factors in determining the responsivity of a detector are Δ , N_A , and W . The peak frequency can be determined by setting the derivative of the responsivity from Eq. 4.1 to zero giving:

$$\frac{d\eta/df}{\eta} - 1/f = 0 \quad (4.2)$$

which depends primarily on Δ and N . In practice, because R is typically calculated numerically, the peak response is determined from a plot of R vs. λ . The primary factor in determining the spectral shape will be the value of Δ chosen for the detector if cavity effects³⁷ are not included. If the optical electric field strength is constant in the device (ignoring the resonant cavity effect), the high frequency end of the response is independent of the Al fraction as η_i becomes almost constant.

The slope of low frequency region of the response depends weakly on x . As a

result, the wavelength for peak response will decrease by roughly the same factor as does λ_0 as Δ is increased. In devices for which the cavity effect is significant the peak response can be shifted significantly from the non-cavity case. Maximum peak response occurs when the device thickness causes the cavity peak to match the position predicted from the non-cavity response curve. Designing the cavity peak to appear near (but not at) the non-cavity response peak will produce a broader response but with a reduced maximum value.

The absolute magnitude of the response can be increased in three ways: increasing 1) the doping, 2) the thickness, and 3) the number of emitter layers. There is a practical limit to increasing the doping density of the emitters due to corresponding increases in the dark current. In addition the doping density should be in a region where the bandgap narrowing is a weak function of it in order to achieve a high uniformity in the devices. The optimum emitter thickness (and number of layers) can be determined by maximizing the device response found by summing up the response of all the layers. For a single emitter structure, the optimum thickness is $\sim 700 \text{ \AA}$, whereas for multi-layer structures (> 20 layers) the optimum thickness of the individual emitter layers is reduced to 150–250 \AA . For example, a 3 μm -thick detector with 1000 \AA -thick barriers would be optimized with 26 layers with 153 \AA -thick emitters. To obtain the optimum thickness for multilayer structures, first, the device thickness is determined to obtain a resonant cavity enhancement at the desired peak wavelength. Next the barrier thickness is determined from the desired dark current limits with thick enough barriers to reduce the tunneling current to the desired level. With these two values fixed the number of layers and the associated emitter thickness is then determined to obtain maximum responsivity.

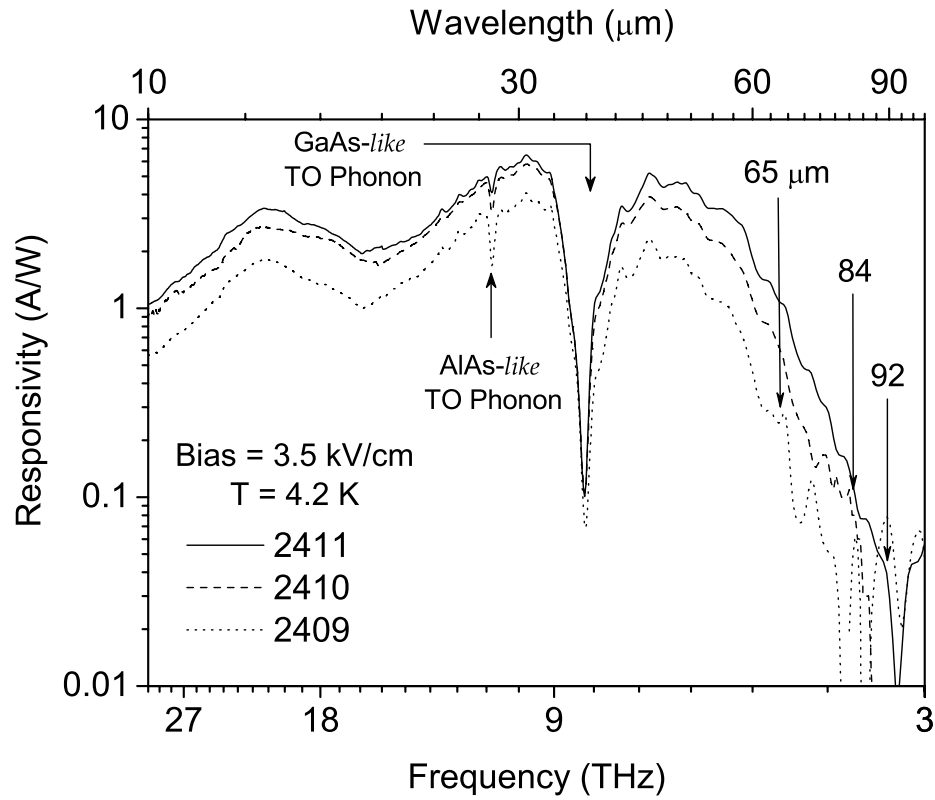


Figure 4.16: Experimental responsivity spectra for detectors 2409, 2410 and 2411 at 3.5 kV/cm obtained at $T = 4.2$ K. The only difference between the detectors is the Al fraction of the barriers, which is $x = 0.02$, 0.01 and 0.005 for 2409, 2410 and 2411, respectively. The data show a decrease in f_0 with decreasing x . The sharp dip near $f = 8$ THz is due to strong interaction between GaAs-like TO-phonons and photons which results in a strong reflection. The small dip at $f = 10.8$ THz is due to AlAs-like TO-phonons in the barrier layers. The arrows indicate both the threshold frequencies of the detectors (x -axis) and the measurement noise levels (y -axis). The different levels are due to dynamic resistance differences of the detectors. The threshold variation; $\lambda_0 = 65$, 84 and 92 μm for detectors 2409, 2410 and 2411, respectively, can be seen clearly.

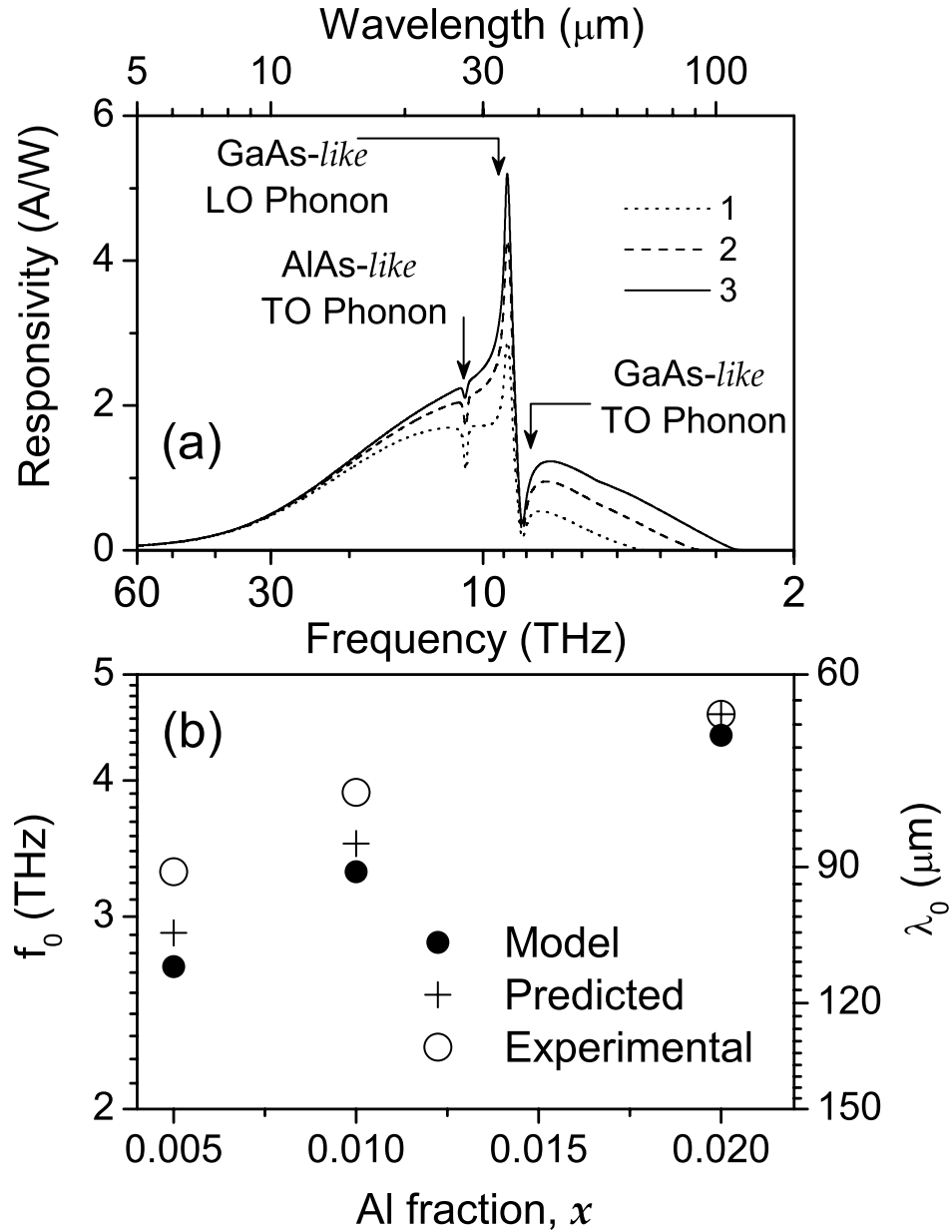


Figure 4.17: (a) Model spectra for 150 Å-thick, $3 \times 10^{18} \text{ cm}^{-3}$ p -type GaAs emitter with $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier forming a single layer detector. Al fractions correspond to the experimental detectors. The model spectra are scaled to match the bias field used for the experimental spectra. (b) The variation in threshold frequency with Al fraction showing a comparison of the spectral threshold, the threshold predicted from the Arrhenius plot, and the model result. The discrepancy between the experimental and predicted results is probably due to small variations in Al fraction.

4.3.1 Experimental

The device structures tested contained 30 periods of $3 \times 10^{18} \text{ cm}^{-3}$ Be:doped 158 Å GaAs emitters (W) and 800 Å $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barriers as shown in Fig. 4.15 giving 31 emitters with the etched top contact acting as an emitter. The Al fraction was varied, with $x = 0.02$, 0.01 and 0.005 for detectors 2409, 2410 and 2411 respectively, to adjust f_0 . The emitters are sufficiently wide that the hole states are not quantized and the expected barrier heights are 18, 13.5 and 11.2 meV respectively. The top and bottom contacts were Be:doped to $1 \times 10^{19} \text{ cm}^{-3}$ with thickness 0.2 and 0.7 μm respectively. The devices were fabricated by etching $400 \times 400 \mu\text{m}^2$ mesas using standard wet etching techniques and then evaporating Ti/Pt/Au ohmic contacts onto the top and bottom contact layers. A $260 \times 260 \mu\text{m}^2$ window was opened through the top contact to provide front illumination to the device. Normal incidence radiation can be absorbed, unlike in QWIPs, allowing simple coupling for incoming radiation.

4.3.2 Results and Discussion

The responsivity was measured using a Perkin-Elmer System 2000 FTIR with a Si composite bolometer as a reference detector. The results obtained for all three detectors at a bias field of 3.5 kV/cm for $T = 4.2 \text{ K}$ are shown in Fig. 4.16. All three detectors have a strong response for frequencies higher than 6 THz. The threshold frequencies are $f_0 = 4.6 \pm 0.1$, 3.9 ± 0.1 , and $3.6 \pm 0.1 \text{ THz}$ (65, 84, and 92 μm) for detectors 2409, 2410 and 2411, respectively, as indicated by the arrows. The responsivity at 10 THz (30 μm) was ~ 4.0 , 5.6, and 6.0 A/W for detectors 2409, 2410 and 2411, respectively. The first

order resonance peaks of the detectors were expected near 8 THz ($\lambda = 42.5 \mu\text{m}$) as the optical path equivalents of their epitaxial stacks match $\sim \lambda/4$ at this frequency. However, the strong photon-phonon interactions around 10 THz, the reststrahlen band for GaAs, results in almost zero throughput of 10 THz radiation, diminishing the first order resonance peaks. As a result, the variation in peak frequency (or wavelength) with x could not be observed for these detectors. The quantum efficiencies at 10 THz ($30 \mu\text{m}$) were 17%, 24%, and 25% for 2409, 2410 and 2411, respectively. The slight difference in response between the detectors is probably due to a combination of the resonant cavity effect and the strong drop in absorption in the reststrahlen region. The noise current of the detectors at $T = 4.2 \text{ K}$ was measured using a spectrum analyzer, giving a current density of $5.2 \text{ pA}/\sqrt{\text{Hz}}$ for detectors 2409 and 2410, and $5.7 \text{ pA}/\sqrt{\text{Hz}}$ for detector 2411. The resulting D^* at $T = 4.2 \text{ K}$ were 3×10^{10} , 4×10^{10} , and $3.8 \times 10^{10} \text{ cm}\sqrt{\text{Hz}}/\text{W}$ for 2409, 2410 and 2411, respectively.

The photoemission of excited carriers is more efficient at lower frequencies,²⁸ with the free-carrier absorption expected to be proportional to f^{-2} , causing the responsivity to increase as frequency decreases. At low frequencies the free-carrier absorption is independent of the frequency, and the responsivity decreases with decreasing frequency due to the reduced internal photoemission. Calculated single layer responsivity will be lower than the experimental responsivity for multilayer detectors as seen by comparing Fig. 4.17(a) with Fig. 4.16. The low frequency response of the device is well described by the model. The experimental data shows increased response at low frequency with decreased Al fraction whereas the high frequency region showed similar response for all three detectors. The peak response and the higher frequency region are strongly influenced by the cavity effect.

Table 4.3: Predicted and measured barrier heights, threshold frequencies and wavelengths from the model, Arrhenius plots, and the spectra for the three detectors showing the threshold wavelength variation with the Al composition in the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barriers. The small variation between the model and spectral values is probably due to deviations of the actual parameters from the design values.

Sample	Δ_{spec} (meV)	Δ_{Arrh} (meV)	Δ_{model} (meV)	f_{spec} (THz)	f_{Arrh} (THz)	f_{model} (THz)	λ_{spec} (μm)
2409	19.1 ± 0.3	19.0 ± 0.2	19.6	4.6 ± 0.1	4.6 ± 0.2	4.7	65 ± 1
2410	14.8 ± 0.2	14.5 ± 0.2	14.3	3.6 ± 0.1	3.5 ± 0.2	3.5	84 ± 1
2411	13.5 ± 0.1	12.0 ± 0.2	11.6	3.3 ± 0.1	2.9 ± 0.2	2.8	92 ± 1

As a result the peak response does not depend significantly on the Al fraction for these devices even though f_0 does show Al dependence, as shown in Fig. 4.16.

The observed f_0 , as well as the values predicted from a modified Arrhenius plot of dark current variation with temperature and the model, for the three detectors are shown in Fig. 4.17(b) and Table 4.3. The threshold frequency decreased (λ_0 increased) as the Al fraction was decreased. The difference in the experimental and model values corresponds to a variation of 1–2 meV in the barrier height which may be due to the deviation of the Al fraction from the nominal value or to the bandgap narrowing. There is also a small difference between the spectral and Arrhenius values for f_0 that is probably due to inelastic scattering of photoexcited carriers before emission¹².

4.3.3 Conclusion

In conclusion, a basic model which can be used to design HEIWIP detectors for desired wavelength ranges is presented in this section. Although the single layer model predicts lower responsivity than observed for frequencies above 10 THz, it gives good results at low frequencies and provides a good starting point for device design. Further studies on the details of the carrier escape process should improve the model in the high frequency range. Based on the model, detectors with different Al fractions were fabricated and the predicted variation in f_0 was observed.

4.4 Free Carrier Absorption in $\text{Al}_x\text{Ga}_{1-x}\text{As}$ Epitaxial Films

Although a threshold wavelength (λ_0) of 92 μm has been realized for HEIWIP detectors based on $p\text{-GaAs}/\text{Al}_x\text{Ga}_{1-x}\text{As}$, further extension is hindered by the practical growth limit of the aluminum fraction (0.005) in MBE grown $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier layers⁴⁹. Even in HIWIP detectors, λ_0 was observed to be limited to $\sim 100 \mu\text{m}$, as increasing the doping density above $2 \times 10^{19} \text{ cm}^{-3}$ would set off depletion of free-holes in the heavy hole band, resulting in a shorter threshold wavelength³⁶. Doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ emitters and GaAs barriers avoid this limit allowing the realization of HEIWIP devices with the λ_0 extended beyond 138 μm ⁴⁹. In this mode, $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ emitters are expected to give a λ_0 of 325 μm . The operation mechanism of these detectors involves FIR absorption in doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ emitters. This makes it important to understand the free carrier absorption in doped AlGaAs epitaxial films, especially in the FIR region where the extended wavelength detectors would operate. Experimental absorption coefficients for $\text{Al}_x\text{Ga}_{1-x}\text{As}$ bulk materials have been reported up to 20 μm ²¹, previously. In this section, experimental results of free carrier absorption in Be:doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial films, in the wavelength range 10–400 μm , are discussed.

4.4.1 Reflectance and Transmittance Measurements

Four Be:doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial films were grown using MBE technique on 520 μm -thick SI-GaAs (100) wafers. The samples were backside polished in order to reduce reflection losses, and to allow transmission measurements. The doping density, thickness, and Al composition of the structures were obtained from SIMS, and are listed in Table 4.4.

Table 4.4: Measured parameters of the p -type $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial films. The 1 μm -thick Be:doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ films were grown on 520 μm -thick SI-GaAs (100) substrates. The doping density (N_p), Al composition, and the epitaxial film thickness were obtained from SIMS. The plasma frequency (ω_p) and the free carrier damping constant (ω_0) were obtained from fitting the reflectance spectra to the model.

Sample	Doping density, N_p $\times 10^{18} \text{ (cm}^{-3}\text{)}$	ω_p $\text{(cm}^{-1}\text{)}$	ω_0 $\text{(cm}^{-1}\text{)}$
$\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$	3.0	279 ± 4	367 ± 17
$\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$	4.7	331 ± 3	339 ± 10
$\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$	7.1	392 ± 2	377 ± 8
$\text{Al}_{0.16}\text{Ga}_{0.84}\text{As}$	4.7	364 ± 6	373 ± 18

Reflectance and transmittance measurements were performed for 3.0×10^{18} , 4.7×10^{18} , and $7.1 \times 10^{18} \text{ cm}^{-3}$ Be:doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films, and for a $4.7 \times 10^{18} \text{ cm}^{-3}$ Be:doped $\text{Al}_{0.16}\text{Ga}_{0.84}\text{As}$ epitaxial film. A piece cleaved from one of the above samples, $3.0 \times 10^{18}\text{-Al}_{0.01}\text{Ga}_{0.99}\text{As}$, was etched down using a wet etchant to expose the GaAs substrate. This was used to obtain the transmission and the reflection spectra for the GaAs substrate. All the measurements were performed at room temperature with a Perkin-Elmer system 2000 Fourier transform infrared spectrometer (FTIR) and a Si composite bolometer. For both the substrate sample and the film/substrate samples, transmittance was measured under a normal incidence geometry, and the reflectance was measured at near normal incidence, $\sim 5^\circ$, using the specular reflectance accessory. For both the reflectance and transmittance measurements, the FTIR instrument resolution was selected at 4 cm^{-1} .

4.4.2 Results and Discussion

The complex permittivity for the GaAs substrate was modeled using Eq. 2.17, and that of the doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial layer was modeled using the Drude theory²⁰ for free carriers in combination with the additive oscillator mode for GaAs-like and AlAs-like phonons²¹, as described in Eq. 2.19. In other words, for the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ film layer, the complex permittivity is given by:

$$\varepsilon(\omega) = \varepsilon_\infty \left[1 - \frac{\omega_p^2}{\omega(\omega + i\omega_0)} \right] + \left(\frac{S_{\text{TO}}\omega_{\text{TO}}^2}{\omega_{\text{TO}}^2 - \omega^2 - i\omega\gamma_{\text{TO}}} \right)_{\text{GaAs}} + \left(\frac{S_{\text{TO}}\omega_{\text{TO}}^2}{\omega_{\text{TO}}^2 - \omega^2 - i\omega\gamma_{\text{TO}}} \right)_{\text{AlAs}} \quad (4.3)$$

The first term in Eq. 4.3 describes the interaction of infrared radiation with free carriers in the doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ film. Here, ε_∞ is the high frequency dielectric constant,

ω is the frequency of incident radiation, ω_0 is free carrier damping constant, and ω_p is the plasma frequency of free carriers. The second and third term describe the interaction of radiation with GaAs-*like* and AlAs-*like* transverse optical (TO) phonons, respectively. Here, ω_{TO} is the TO-phonon frequency, γ is TO-phonon damping constant, and S is the TO-phonon oscillator strength.

The model reflectance spectra of the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ films on the GaAs substrates, and the GaAs substrate alone were generated using the transfer matrix method described in Chapter 2. The complex refractive index (\tilde{n}) of the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ film layer and the GaAs substrate were calculated using Eqs. 4.3 and 2.17, respectively. The phonon parameters of the GaAs substrate were obtained by fitting the reflectance model to its experimental spectrum. These parameters were fixed in obtaining the plasma frequency and the free carrier damping constant of the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epitaxial film layers. The model absorption coefficient, α , of the film was calculated using $\alpha = 2(\omega/c)\text{Im}[\tilde{n}(\omega)]$, where $\text{Im}[\tilde{n}(\omega)]$ is the extinction coefficient equal to the imaginary part of $\tilde{n}(\omega)$ and c is the speed of light.

The experimental and model reflectance spectra of the $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ film samples at room temperature are shown in Fig 4.18. The sharp peak in the spectra at 8.1 THz (37 μm) is due to the interaction of radiation with GaAs-*like* TO-phonons, and the peak at 10.7 THz (28 μm) is due to AlAs-*like* TO phonons. These values are similar to those reported earlier^{21, 53}. The values of Drude parameters obtained from the fitting are listed in Table 4.4. The plasma frequency increases with the doping density. Although the doping density of $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ and $\text{Al}_{0.16}\text{Ga}_{0.84}\text{As}$ epitaxial films are same ($4.7 \times 10^{18} \text{ cm}^{-3}$), their plasma frequencies are slightly different. However, the value is located between the samples

with doping densities of 3.0×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$. This discrepancy could be due to the effects of high Al composition and AlAs-*like* TO phonons. The reflectance spectrum for $\text{Al}_{0.16}\text{Ga}_{0.84}\text{As}$ clearly show that the high Al fraction has increased the strength of the AlAs-*like* TO-phonons while reducing the strength of the GaAs-*like* TO-phonons.

Reflectance spectra for all samples showed an intensity oscillation in the FIR range. Decreasing the FTIR instrument resolution to 1 cm^{-1} resolved the oscillations fully and the their frequency was found to be of $2.7 \pm 0.1 \text{ cm}^{-1}$. This matches the frequency of Fabry-Pérot interference caused by the $520 \text{ }\mu\text{m}$ -thick GaAs substrate. Therefore, the experimental absorption coefficient of the films were calculated using a model (Eqs. 4.4 and 4.5) that includes the multiple reflections within the GaAs substrate. However, the multiple reflections within the film layer were not included due to the low reflectivity, ~ 0.03 , of the $\text{Al}_x\text{Ga}_{1-x}\text{As}/\text{GaAs}$ interface.

$$\alpha_{\text{Film}}d_{\text{Film}} + \alpha_{\text{Sub}}d_{\text{Sub}} = \sinh^{-1} \left[\frac{(1 - R_{\text{sample}})^2 - T_{\text{sample}}^2}{2T_{\text{sample}}} \right] \quad (4.4)$$

$$\alpha_{\text{Sub}}d_{\text{Sub}} = \sinh^{-1} \left[\frac{(1 - R_{\text{Sub}})^2 - T_{\text{Sub}}^2}{2T_{\text{Sub}}} \right] \quad (4.5)$$

The experimental free carrier absorption coefficient (α) spectra for the $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films with different (Be)doping densities are shown in Figs. 4.19(a) and 4.19(b). The dashed curve shows α for $3 \times 10^{18} \text{ cm}^{-3}$. The solid and the dotted curves are for 4.7×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$, respectively. These absorption coefficients were obtained using the experimental data for transmittance and reflectance in Eqs. 4.4 and 4.5. The transmittance (for both the GaAs substrate alone and the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ samples) in the reststrahlen region, shown by a break in Fig. 4.19, falls below the spectrometer noise level

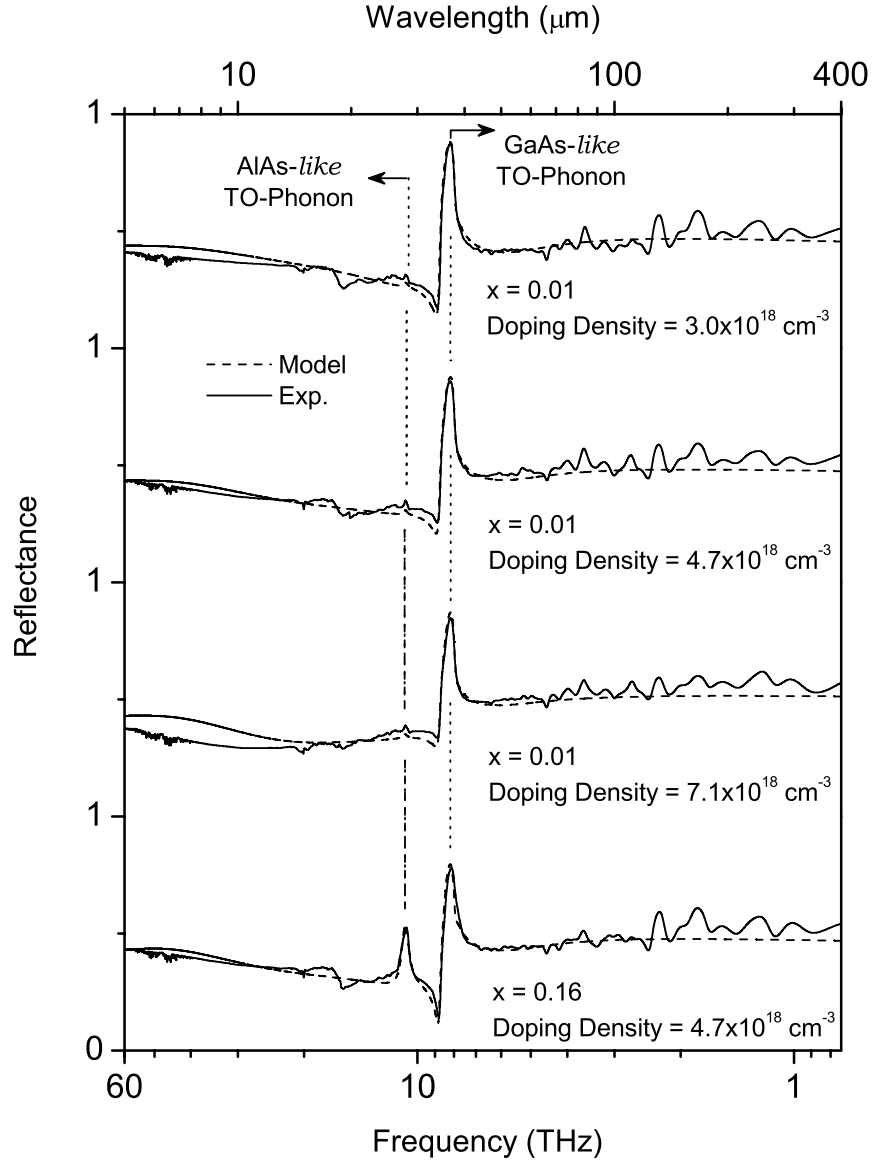


Figure 4.18: Experimental (solid line) and model (dashed line) reflectance spectra for the Be-doped epitaxial films grown on 520 μm -thick GaAs SI-substrates. The sharp peaks at $\sim 37 \mu\text{m}$ are due to the interaction of radiation with GaAs-like TO phonons, and the arrows at $\sim 28 \mu\text{m}$ point to small peaks due to interaction with AlAs-like TO phonons. The strength of AlAs-like phonons increases with Al composition as shown. The ripples towards the FIR matches the Fabry-Pérot interference in the SI-substrate.

due to the high reflectance caused by phonons of both the GaAs substrate and $\text{Al}_x\text{Ga}_{1-x}\text{As}$ samples. This would blow up the calculation of absorption coefficient for AlGaAs in this region. The absorption curves correspond to room temperature. No major changes in α are expected with temperature due to very low temperature coefficients associated with the relevant optical constants for $\text{Al}_x\text{Ga}_{1-x}\text{As}$ ²¹. Figure 4.19(b) shows α increasing from around 3×10^3 to $4.7 \times 10^3 \text{ cm}^{-1}$ as doping density increases from 3×10^{18} to $7.1 \times 10^{18} \text{ cm}^{-3}$. Although a $\sim \lambda^2$ dependency is observed in the shorter wavelength region, the free carrier absorption in Be:doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films was found to be almost independent of the wavelength in the range 100–400 μm .

The absorption coefficient spectra for $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epilayers with different Al compositions (x) and a (Be)doping density of $4.7 \times 10^{18} \text{ cm}^{-3}$ are shown in Fig. 4.20(a). The absorption coefficient has decreased from $\sim 3.5 \times 10^3$ for $x=0.01$ to $3 \times 10^3 \text{ cm}^{-1}$ for $x=0.16$, while being wavelength independent in the range 100–400 μm . The threshold wavelength in GaAs emitter/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier HEIWIP devices could only be extended so far to a maximum λ_0 of 92 μm , and this was ascribed to the minimum Al fraction growth limit of ~ 0.005 in MBE. However, in HEIWIP detectors with doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ emitters, λ_0 increases with x avoiding the limit imposed by Al fraction control. Using a detector with doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ layers as emitters a λ_0 of $\sim 325 \mu\text{m}$ could be obtained for a structure with intrinsic GaAs barriers. Further, calculations show that the FIR absorption coefficient in the range 5–400 μm decreases only by $\sim 0.5 \%$, when a $4.7 \times 10^{18} \text{ cm}^{-3}$ Be:doped GaAs emitter is replaced with an $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ emitter of similar doping density. Hence, FIR absorption, almost similar to that in GaAs can be obtained using emitter layers with

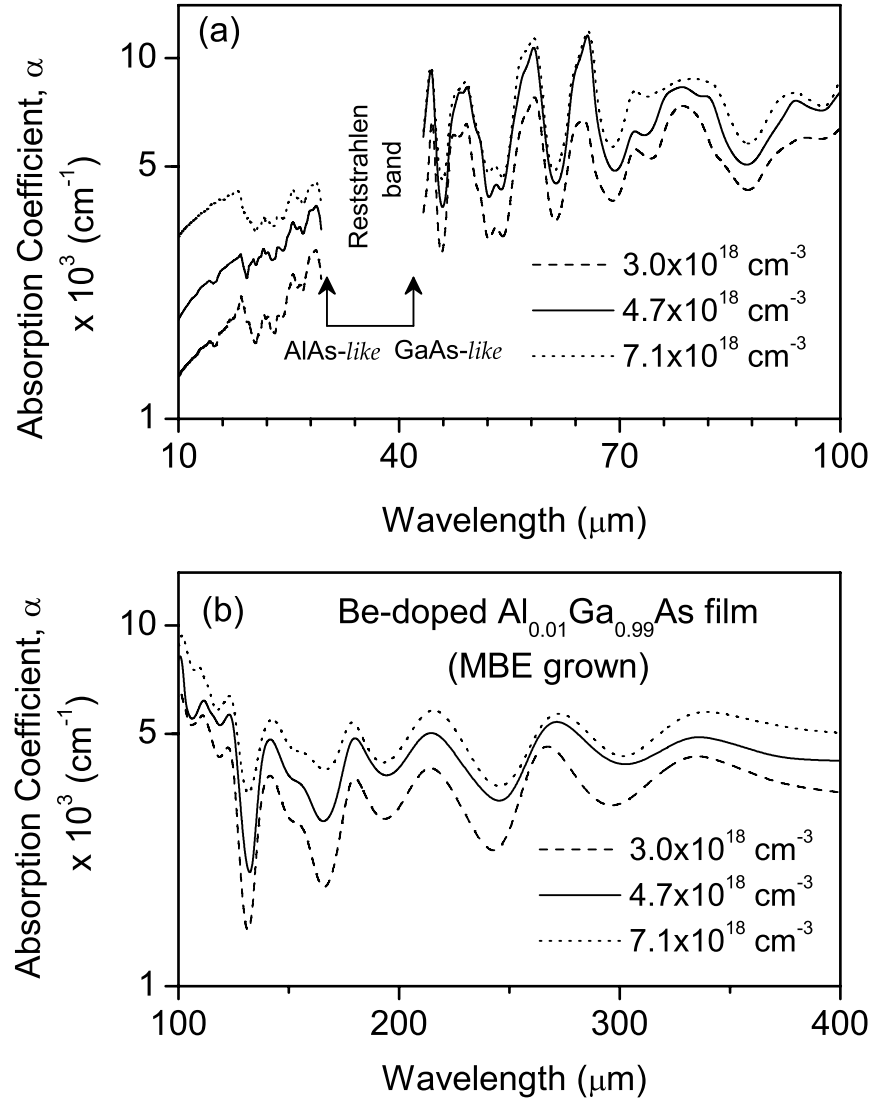


Figure 4.19: (a) Experimental absorption coefficient (α) in the range 10–100 μm for Be-doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ MBE-grown epitaxial films. the measurements were done at room temperature. The dashed, solid and dotted curves show α for films with doping density 3×10^{18} , 4.7×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$, respectively. The region shown by the break corresponds to the combined reststrahlen band of GaAs-like and AlAs-like TO-phonons of $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$. (b) The absorption coefficient is almost independent of wavelength in the FIR range 100–400 μm . The absorption coefficient in this region are 3×10^3 , 3.5×10^3 and $5 \times 10^3 \text{ cm}^{-1}$ for $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films with doping density 3×10^{18} , 4.7×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$, respectively.

lower Al fractions. Figure 4.20(b) shows a sub-linear relationship between α and the acceptor doping density (N_A) for $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epilayers with similar doping density. The absorption coefficient values represent the average in the range 100–200 μm . The α is proportional to $\sim p^{0.5}$. Therefore, the FIR absorption in $\text{Al}_x\text{Ga}_{1-x}\text{As}$ film layers can be expected to increase with the doping density.

The model and the experimental absorption coefficient spectra for the $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epilayer with $7.1 \times 10^{18} \text{ cm}^{-3}$ doping density are shown in Fig. 4.21. The peak in the experimental curve around 3 μm is due to the photoexcitation from heavy and light hole bands to split off band of $\text{Al}_x\text{Ga}_{1-x}\text{As}$, which is not included in the model.

4.4.3 Conclusion

Free carrier absorption in doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ films, grown using the MBE technique on SI-GaAs (100) substrates, was investigated. Free carrier absorption for three different hole densities with the same Al fraction and for two different Al fractions with the same doping density was studied. Experimental absorption coefficients were obtained from data using a model that includes multiple reflections in the substrate wafer. In the 100–400 μm range, 3×10^{18} , 4.7×10^{18} and $7.1 \times 10^{18} \text{ cm}^{-3}$ Be:doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epilayers have absorption coefficients of $\sim 3 \times 10^3$, 3.5×10^3 and $5 \times 10^3 \text{ cm}^{-1}$, respectively. Here, the magnitude of the absorption is found to be almost independent of the wavelength. This allows replacing doped GaAs emitters in HEIWIP detectors with p -type $\text{Al}_x\text{Ga}_{1-x}\text{As}$ layers with $x < 0.017$. This facilitates the threshold wavelength extension of HEIWIP detectors beyond the 92 μm limit as discussed before.

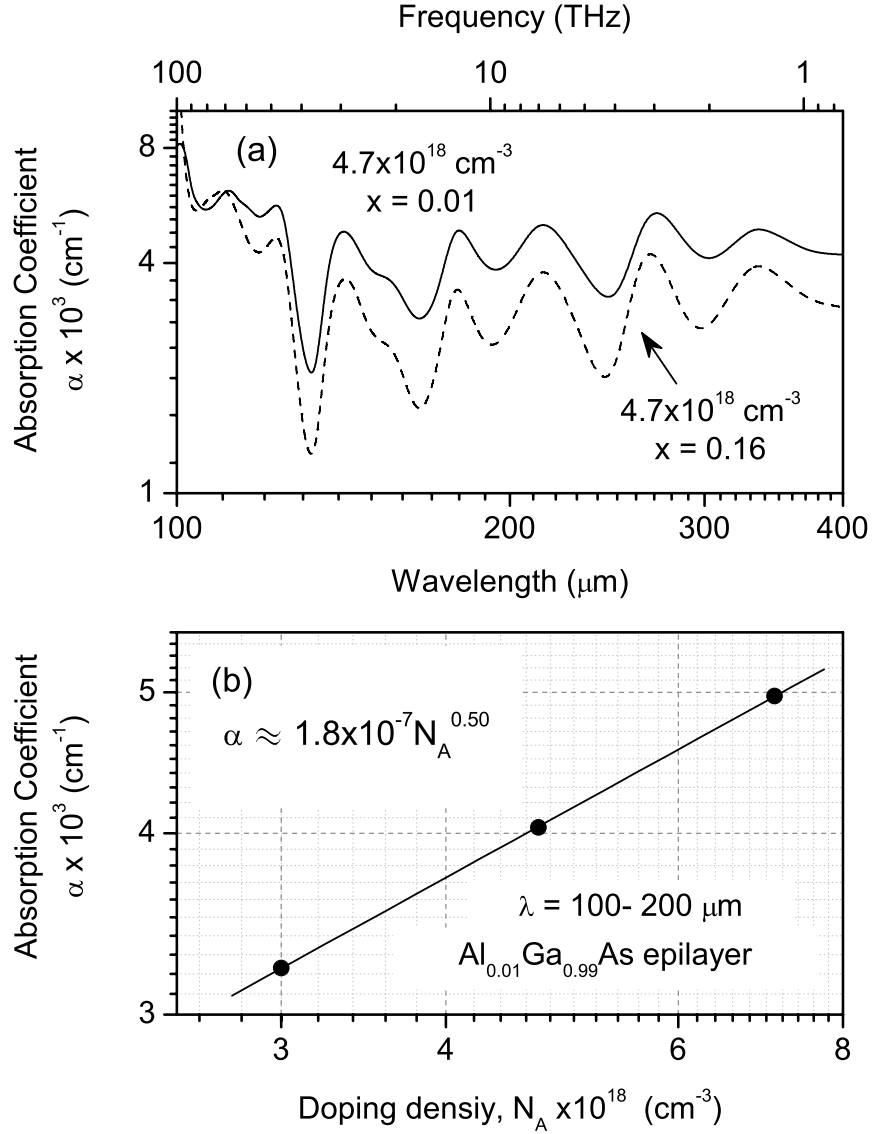


Figure 4.20: (a) Experimental absorption coefficient (α) in the range 100–400 μm for Be-doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ epilayers with different Al composition. The curve was derived from room temperature reflection and transmission spectra. Free carrier absorption is found to be almost independent of wavelength in the 100–400 μm range. The value of α has decreased from $\sim 3.5 \times 10^3$ for $x = 0.01$ to $\sim 3 \times 10^3 \text{ cm}^{-1}$ for $x = 0.16$. (b) Sub-linear relationship of the free carrier absorption coefficient with acceptor (Be) doping density, $\alpha \propto p^{0.5}$, for $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epitaxial films. The values represent the average in the range 100–200 μm .

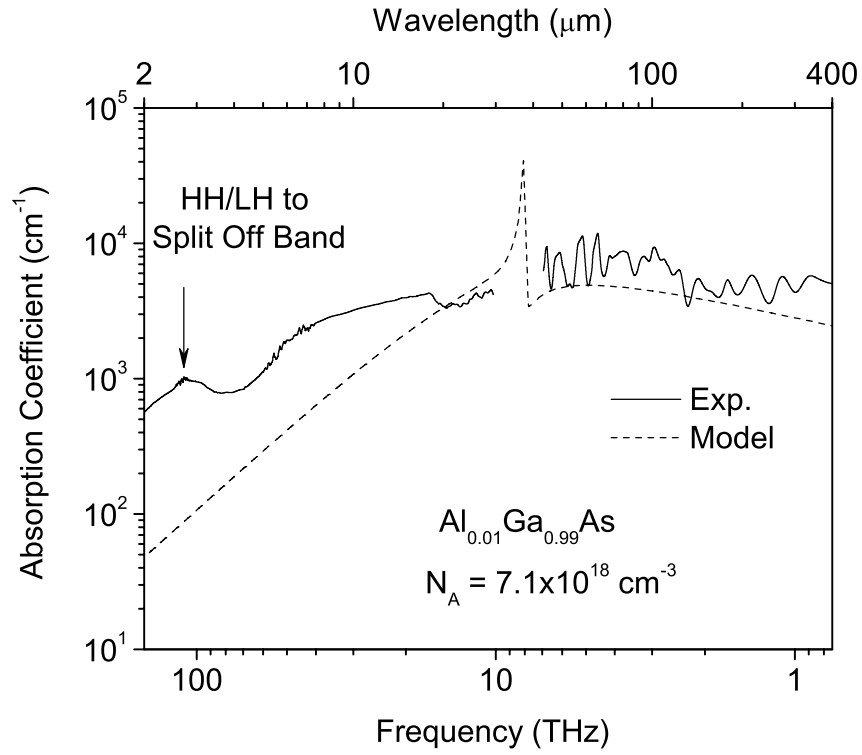


Figure 4.21: The model and experimental free hole absorption coefficient for a $7.1 \times 10^{18} \text{ cm}^{-3}$ Be-doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ epilayer from 2 to 400 μm at room temperature. The peak around 3 μm is due to carrier transitions from the heavy and light hole bands to the split off band of $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$. The model does not include this mechanism.

In conclusion, free carrier absorption in the range 10–400 μm has been investigated. The absorption coefficient for $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ is almost the same for GaAs, and this facilitates the use of thin acceptor doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ absorber layers as emitters in threshold extension of HEIWIP FIR detectors. The sub-linear relationship between the absorption coefficient and the doping density would be useful in designing HEIWIP detectors. Because the variation of α with the Al fraction is not significant for small Al fractions, the absorption quantum efficiency for HEIWIP detectors with different threshold wavelengths will not vary significantly.

4.5 Threshold Extension Using $\text{Al}_x\text{Ga}_{1-x}$ Emitters

Although a threshold frequency (f_0) of ~ 3.3 THz has been realized⁴⁹ in heterojunction interfacial workfunction internal photoemission (HEIWIP) detectors with GaAs emitters and $\text{Al}_{0.005}\text{Ga}_{0.995}\text{As}$ barriers, further extension is restricted by the Al fraction growth accuracy and the transition from alloy to iso-electronic doping behavior when to $x \leq 0.005$ in MBE grown structures. A detector without any Al (i.e. $x=0$) will give rise to a homojunction interfacial workfunction internal photoemission (HIWIP) detector where f_0 was observed to be limited to ~ 3 THz due complications associated with doping density above $2 \times 10^{19} \text{ cm}^{-3}$ ³⁶. However, doped $\text{Al}_x\text{Ga}_{1-x}\text{As}$ emitter and GaAs barrier HEIWIP detectors overcome these limits allowing the realization of f_0 down to ~ 1 THz and lower. Experimental results have shown that the free carrier absorption coefficients in Be:doped $\text{Al}_{0.01}\text{Ga}_{0.99}\text{As}$ are almost the same as those in GaAs with similar doping density⁵⁴. This promises similar photon absorption probabilities as in previously reported Be:doped GaAs emitter/ $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier HEIWIP detectors⁵⁰.

The heterojunction in HEIWIP detectors allows reducing f_0 by tuning the band offset using Al fraction⁴⁹ without increasing the doping density as in homojunction detectors³⁵. A low enough doping density maintains a lower dark current⁵⁰, which improves the detectivity. Compared to the Ga doped Ge detectors⁵ currently used in this range, array fabrication will be much easier with HEIWIP detectors as the application of mechanical stress to individual pixels of the array is not needed. The other major alternative, bolometers, have a very low response speed ($< 1000\text{Hz}$), whereas the HEIWIP detectors have transit-time limited speeds on the order of GHz for a bias field of 2 kV/cm . The quantum well photode-

tectors (QWIPs) have recently extended the f_0 to ~ 5 THz⁵⁵. However, it is still below the response region of HEIWIP detectors and in QWIPs, and decreasing the frequency further below this value requires the doping density to be decreased to reduce dark current, but this lowers the absorption and therefore the sensitivity. Recently, a three terminal device operating at 2.5 THz was reported⁵⁶.

4.5.1 Experimental

The $\text{Al}_{0.005}\text{Ga}_{0.995}\text{As}/\text{GaAs}$ HEIWIP detector structure (V0207) was grown using MBE technique on a SI-GaAs (100) substrate. The structure consists of 10 periods of $3 \times 10^{18} \text{ cm}^{-3}$ Be:doped 500 Å-thick $\text{Al}_{0.005}\text{Ga}_{0.995}\text{As}$ emitter and 2000 Å-thick GaAs barrier sandwiched between two contacts. The top and bottom contacts are $1 \times 10^{19} \text{ cm}^{-3}$ Be:doped 500 Å and 7000 Å-thick $\text{Al}_{0.005}\text{Ga}_{0.995}\text{As}$ layers, respectively. The sample was processed by delineating square mesa elements of four different areas from 400×400 to $1000 \times 1000 \mu\text{m}^2$ by wet etching techniques. As the top contact is only 500 Å-thick, no etching was done to reduce the thickness, therefore; the top contact was included as a part of the first emitter in calculating hot carrier injection for the model response. This is different from the previous devices which used thicker top contacts, and required a window to be etched for front illumination³⁵. The ohmic contacts were formed by depositing Ti/Pt/Au.

4.5.2 Results and Discussion

A partial band diagram of the active region of HEIWIP detectors with intrinsic barrier is shown in Fig. 4.22(a). The basic operation⁵⁷ is a three stage process: free carrier absorption, internal photoemission, and collection of photoemitted carriers. As free

carrier absorption involves initial and final energy states within the same continuum, the response of HEIWIP detectors is inherently broad band and not limited by absorption at longer wavelengths. The zero response threshold frequency, f_0 , is introduced only in the photoemission stage. The internal photoemission of the carriers is characterized by the interfacial workfunction, Δ , which corresponds to the energy difference between the bottom of the barrier (for a p -HEIWIP) and the Fermi level in the emitter. Therefore, the threshold frequency of the detector is $f_0 = 0.242\Delta$ with f_0 in THz and Δ in meV.

For HEIWIP detectors with undoped barriers, Δ has two contributing factors: (i) the band gap offset due to the difference in materials between the emitter and barrier Δ_x ; (ii) the band gap narrowing⁵⁸ due to the doping in the emitter layers Δ_d , giving a total $\Delta = \Delta_d + \Delta_x$ ($\Delta_x < 0$) as shown in Fig. 4.22(a). In the range $N_A \sim 1-8 \times 10^{18} \text{ cm}^{-3}$, Δ_d does not vary significantly with doping, whereas Δ_x can be varied by adjusting the Al fraction of the emitters. However, as shown in Fig. 4.22(b) bending originated from the effective residual doping in the barriers could modify Δ such that $\Delta = \Delta_d + \Delta_x + \Delta_b$. The effective doping may be due to the compensation of dopants in the barriers. The n -dopants will be fully ionized, whereas the p -dopants will be only partially ionized, giving a net positive doping density. For an unbiased detector, $\Delta_b \simeq 13 \text{ meV}$ for a residual doping of $1.1 \times 10^{15} \text{ cm}^{-3}$. As Δ_b decreases with the bias field, f_0 can be tailored with the ideal minimum corresponding to an intrinsic barrier.

Dark and background photocurrent ($T_{\text{BG}} = 300 \text{ K}$ with $\text{FOV} = 60^\circ$ at the cold stop) of the devices were measured using a Keithly source meter with the sample mounted on the cold finger of a liquid He flow cryostat. The spectral response was measured with

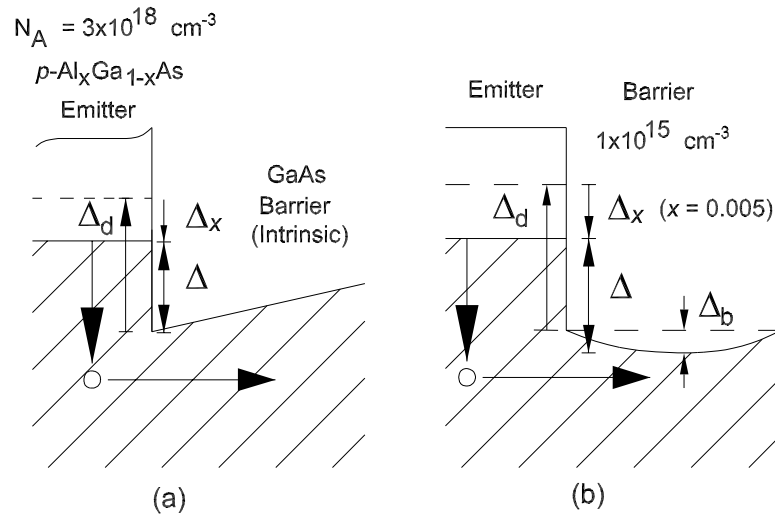


Figure 4.22: (a) A partial band diagram of the active region of the HEIWIP device, with an intrinsic barrier under bias showing the contributions to the workfunction from the band gap narrowing (Δ_d) in the doped emitter and the Al_xGa_{1-x}As/GaAs valence band offset ($\Delta_x < 0$). The dashed line indicates where the location of the valence band edge in the barrier would be if it were GaAs. Here, $\Delta = \Delta_d + \Delta_x$ where x is the Al fraction. (b) Band bending due to residual doping in the barrier. For zero bias field Δ_b is ~ 13 meV. Because Δ_b is a function of the bias field, for a given bias $\Delta = \Delta_d + \Delta_x + \Delta_b$ and Δ decreases with the bias.

4 cm^{-1} resolution using a Perkin-Elmer system 2000 Fourier-transform infrared (FTIR) spectrometer with a Si composite bolometer as the reference. An SR-785 two-channel fast Fourier transform signal analyzer coupled to a SR-560 high impedance low-noise voltage pre-amplifier was used for noise density measurements with the sample immersed in liquid He. Intrinsic dark noise levels of the detector were found by measuring four noise current components⁵⁹.

The dark currents at different temperatures were used to obtain Δ through Arrhenius plots. The variation of Δ with the bias field for three devices with different electrical areas are shown in Fig. 4.23. The uniform workfunction is an advantage for detector arrays in terms of spectral shape and detectivity. The Δ at zero bias is ~ 17 meV for all the mesas, decreasing to 10.5 meV at 2 kV/cm. This strong bias dependence is a signature for the presence of residual doping in the barriers⁶⁰ A residual doping of $1.1 \times 10^{15} \text{ cm}^{-3}$ was estimated from the Be counts of SIMS. The small periodic spikes seen on the curves may be due to resonant tunneling through defects in the structure. This is possible because of the low temperature maintained during the growth to minimize the expected Be diffusion.

The variation of responsivity with the bias field at $T = 4.8$ K is shown in Fig. 4.24. For frequencies > 6 THz, the responsivity increases with the field with a maximum of 9 A/W at 1.5 kV/cm. Although a further increase in the field decreases the higher frequency response, the lower frequency (< 6 THz) response increases due to the reduction in band bending, hence Δ , with the field (see Fig. 4.23). As expected, Δ_b decreases with the field, thus increasing the threshold to 2.3 THz for a bias field of 2.0 kV/cm. The semi-log scale of Fig. 4.24 clearly shows the variation of f_0 with the applied field. Although further reduction

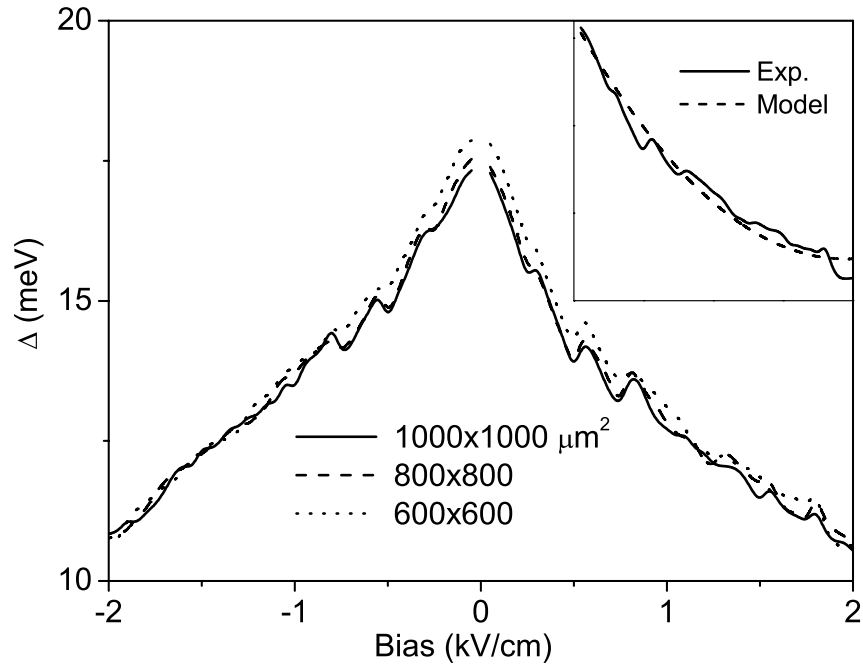


Figure 4.23: Variation of the workfunction, Δ , with the bias field for three mesas with different electrical areas. The workfunctions at different bias fields were obtained using Arrhenius plots. The zero bias workfunction is ~ 17 meV for all the mesas. Inset shows the experimental and model variation of barrier height for the device with $1000 \times 1000 \mu\text{m}^2$ electrical area. The variation with the bias is due to band bending caused by space charge in the barrier layer.

in f_0 was expected with increasing bias, the increasing noise floor of the detector limits it to 2.3 THz. The threshold frequencies along with the figures of merit for different bias fields are shown in Table 4.5. The threshold frequency was obtained using the mean of multiple measurements and the noise floor under dark condition for the given bias field. For $f_0 = 2.3$ THz, the best performance of the detector is at 9.6 THz ($31 \mu\text{m}$), where the responsivity, quantum efficiency, and dark current limited specific detectivity are 7.3 A/W, 29% and $5.3 \times 10^{11} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, respectively.

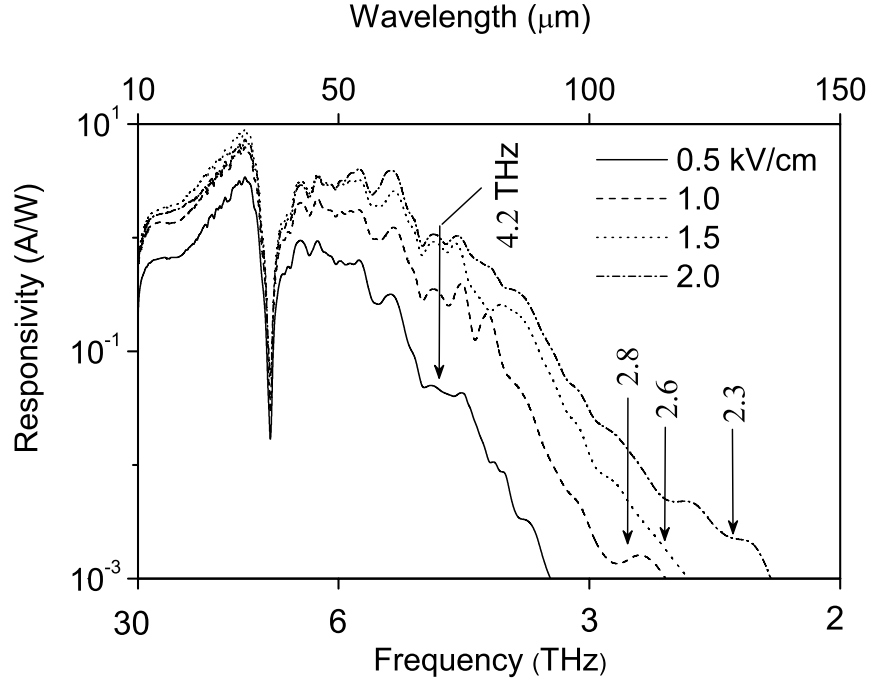


Figure 4.24: The variation of responsivity with applied field for detector V0207 at $T = 4.8$ K. The peak responsivity, 9 A/W at 9.6 THz, was obtained at 1.5 kV/cm. The increase in response with the field around f_0 is due to threshold shift with the bias. The sharp dip at ~ 8 THz is due to the interaction of radiation with GaAs-like TO phonons. The bias field decreases the effective work function pushing f_0 towards 2 THz with the increasing field.

Table 4.5: Variation of threshold frequency (wavelength) with the applied bias field for detector V0207. The increasing field decreases the band bending in the barrier which decreases the threshold frequency.

Bias field (kV/cm)	λ_0 (μm)	f_0 (THz)	R_P (A/W)	η_P (%)	D_P^* ($\text{cm}\sqrt{\text{Hz}}/\text{W}$)	D_{Shot} ($\text{cm}\sqrt{\text{Hz}}/\text{W}$)
0.5	71.2 ± 0.3	4.21 ± 0.02	3.4	13	—	—
1.0	108.1 ± 0.6	2.77 ± 0.02	6.3	25	—	—
1.5	115 ± 6	2.6 ± 0.1	9.0	36	1.5×10^{13}	1.7×10^{13}
2.0	128 ± 9	2.3 ± 0.2	7.3	29	5.3×10^{11}	1.5×10^{12}

A dark current limited peak detectivity of $1.5 \times 10^{13} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, which is close to the shot noise limit, was obtained at the bias field of 1.5 kV/cm at $T = 4.2 \text{ K}$. Because the differential resistance of the detector at bias fields $< 1.5 \text{ kV/cm}$ is in the range of several Giga ohms, determination of the intrinsic noise level of the detector is beyond our instruments limit. At very high impedance, the measured noise was limited by the Johnson level of the load resistor. A BLIP temperature of 20 K for a 0.15 kV/cm bias field was recorded.

Model responsivity spectra for V0207 ($N = 10$) and two similar structures with different emitter/barrier units, $N = 40$ and 50 , are shown in Fig. 4.25. The first order cavity for these structures are at $f = 5.5$ and 4.9 THz , respectively. As the total thickness of the detector increases, higher order cavity peaks at higher frequencies start appearing in the responsivity spectra. The model response for $N=50$ shows the cavity peaks of orders 3, 7, 9,... whereas the order 5 falls in the reststrahlen band of GaAs²¹. A band bending contribution of $\Delta_b = 3 \text{ meV}$, corresponding to a residual doping of $1.1 \times 10^{15} \text{ cm}^{-3}$ in the barrier and a bias field of $\sim 2 \text{ kV/cm}$ was considered in the model. Although the overall response improves with increasing N , responsivity below $\sim 3 \text{ THz}$ does not increase for $N > 40$. For the structure with $N=40$, the first cavity peak is at 5.5 THz , giving a responsivity $> 10 \text{ A/W}$ (quantum efficiency, $\eta > 25\%$) at 6 THz and 2.4 A/W ($\eta = 3\%$) at 3 THz .

The inset to Fig. 4.25 shows experimental responsivity curve for 2.0 kV/cm bias field along with the model ($N=10$). The best fit to the experimental response for 2.0 kV/cm yields an optical gain of 2. The two arrows indicate the positions of cavity peaks, the shoulder at 21.4 THz is caused by the third order cavity peak, and the first order peak at

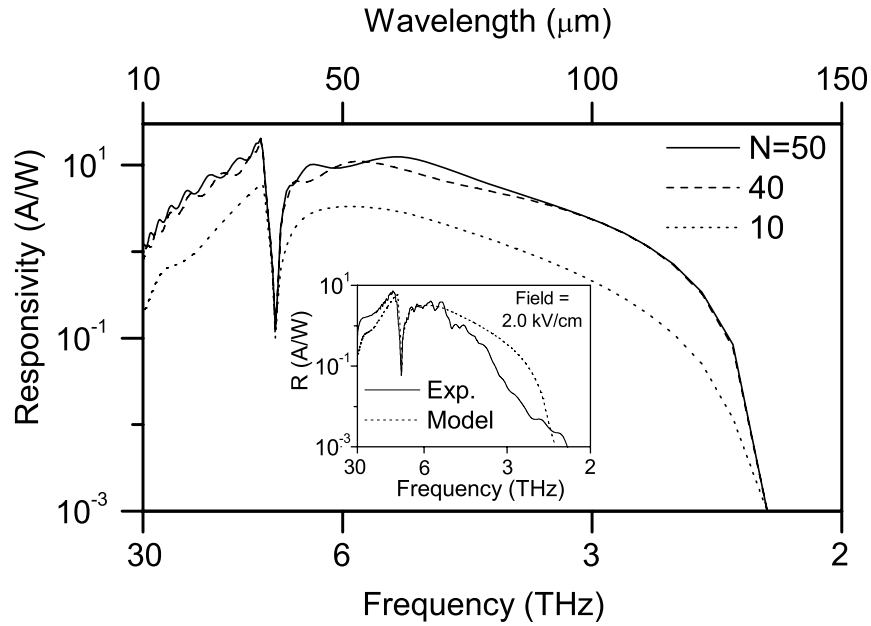


Figure 4.25: Calculated responsivity spectra for structures with similar parameters as in V0207 and emitter/barrier units of $N=10, 40$, and 50 are shown. The responsivities are for a bias field of ~ 2.0 kV/cm. The detector V0207 is not cavity optimized, whereas the model structures with $N=40$ and 50 are optimized for 5.6 and 4.9 THz, respectively. The cavity peaks of order $3, 7, 9\dots$ are seen, whereas order 5 (7.7 THz) falls within the reststrahlen band of GaAs. The inset shows the experimental responsivity of detector V0207 for a bias field of 2 kV/cm with the calculated responsivity with an optical gain of 2 . The arrows indicate the cavity peaks of the detector. The shoulder at $14\text{ }\mu\text{m}$ on both the experimental and the model curves corresponds to the third order cavity peak with the first order (7.3 THz) peak falling within the reststrahlen band of GaAs.

7.3 THz falls in the reststrahlen band. A reflection measurement performed on a piece from the same wafer as the V0207 detector showed higher order cavity peaks above 30 THz. The other small peaks throughout the experimental curve are due to the 620 μm -thick substrate interference effect, which is not considered in the model.

4.5.3 Conclusion

A heterojunction interfacial workfunction internal photoemission (HEIWIP) detector with a threshold frequency (f_0) of 2.3 THz ($\lambda_0 = 128 \mu\text{m}$) was demonstrated. The threshold limit of ~ 3.3 THz (92 μm) due to the Al fraction being limited to ~ 0.005 , in order to avoid growth difficulties in controlling low compositions and transition from alloy to iso-electronic doping behavior, was surpassed using AlGaAs emitters and GaAs barriers. The peak values of responsivity, quantum efficiency, and the specific detectivity at $f = 9.6$ THz and 4.8 K for a bias field of 2.0 kV/cm are 7.3 A/W, 29% and $5.3 \times 10^{11} \text{ cm}\sqrt{\text{Hz}}/\text{W}$, respectively. A BLIP temperature of 20 K for FOV = 60° was recorded at a bias field of 0.15 kV/cm. The f_0 could be further reduced towards ~ 1 THz regime ($\sim 300 \mu\text{m}$) by adjusting the Al fraction to offset the effect of residual doping, and/or lowering the residual doping in the barrier, effectively lowering the band bending.

An AlGaAs emitter HEIWIP detector with a threshold frequency of 2.3 THz (wavelength of 128 μm) was demonstrated. The threshold can be increased further by compensating the band bending with the emitter Al fraction. AlGaAs emitter GaAs barrier devices will be grown considering this effect to increase the detector threshold close to ~ 1 THz region.

Chapter 5

Hot Electron GaAs/Al_xGa_{1-x}As Heterostructure Design for Pulsing

5.1 Introduction

Spontaneous pulsing was first observed in circuits containing $p^+ - n - n^+$ diodes driven by either a constant voltage source⁶¹ or a constant current source⁶². Since then, there were few other pulsing heterojunction structures reported sporadically. Size and temperature are two of the major drawbacks in $p^+ - n - n^+$ diodes, limiting their applications in real situations. On the nano-structure front, multi-quantum well GaAs/AlGaAs device structures were shown to pulse at room temperature. Although these were occasional successes, a major breakthrough of the underlying physics principles is needed to achieve reproducibility of such devices. On the other hand spontaneous pulse trains at room temperature can be readily achieved using a silicon controlled rectifier (SCR) coupled to an

external capacitor. Their bulky nature, along with their high power consumption, will limit most practical applications. The similarity of these pulses to action potentials in neurons can be used to emulate neural networks such as the detection of transient optical signals in a horseshoe crab eye⁶³. As an initiative towards obtaining a hot electron device pulsing at room temperature, we modeled the switching mechanism of a device based on the mature GaAs/AlGaAs material system. The model is based on switching arising from barrier breakdown at high field⁶⁴. Nevertheless, at present, one of the GaAs/AlGaAs multiple quantum well device structure shows pulsing up to a device temperature of 11 K^{65, 66}. This is slightly higher in temperature and very small in size compared to the previous Silicon $p^+ - n - n^+$ diodes. However, pulsing ceases above this temperature, limiting the practical applications that require room temperature operation. For example, a photoreceptor channel in a biological retina can be simulated using two pulsing outputs through a filter circuit⁶⁷.

Increasing the temperature of pulsing devices to room temperature, if not close to it, has been the research focus of several leading scientists in the past several years^{68, 64, 69}. The device used in this study is a HHED, a two terminal device consisting of a GaAs and AlGaAs layers. In order to have pulsing (or at least oscillations), the device must exhibit an S-shaped negative differential conductivity. Although this is a necessary condition, this type of switching in conductivity would not guarantee that a device would pulse. The shape of the S-region should be such that it should allow the device to fall back to the low conductivity region once it is switched to the high conductivity region of the negative differential resistance (NDR). The switch between the two regions is based on electrical

bi-stability between the tunneling and thermionic emission across the barrier of the device. First ideas of the physical mechanism in a HHED was discussed in Ref. 70. Here, we expand the HHED mechanism by using a more accurate trapezoidal/triangular coherent-tunneling mechanism. Unlike in the WKB approach for the above mechanism, the transmission probability function varies smoothly across the barrier energy, giving a more precise value for the injection current. The model also takes into account the difference arising in the effective mass when the carriers cross into the alloy barrier (especially when the alloy fraction is considerable), and the tunneling current contribution from the bound states of the two dimensional electron gas (2-DEG) space charge well.

5.2 Switching in Heterostructure Hot Electron Devices

In this section, carrier transport mechanisms, such as the transition of hot carrier population from the tunneling current regime to the thermionic regime, are discussed, as these are the phenomena believed to be responsible for pulsing observed in hot electron diodes. Although the theories of thermionic and field emission are very well understood in the frame of linear transport models for metal/vacuum and semiconductor/vacuum interfaces⁷¹ the nonlinear characteristics⁶⁴ of a HHED have been modeled in the frame of a non-linear transport mechanism⁷². Generally, for a HHED with a wide gap barrier, used to suppress thermal emission at the operating temperature of the HHED, the tunneling current predominates at low temperatures and high bias fields, whereas the thermionic emission predominates at high temperatures and low bias fields. In other words, the dominant transport mechanism switches from tunneling to thermal emission at high field with

the temperature of the HHED fixed. The tunneling regime has a high series resistance which drops significantly in the thermal emission regime.

Evolution of the electron temperature in a 2-DEG was modeled for a structure based on the GaAs/Al_xGa_{1-x}As material system. The model structure has a $1 \times 10^{17} \text{ cm}^{-3}$ doped *n*-GaAs layer and an undoped Al_{0.46}Ga_{0.54}As barrier layer sandwiched between two highly doped ($1 \times 10^{18} \text{ cm}^{-3}$) *n*-type GaAs contact layers on a *n*-type substrate. The partial band diagrams of the structure before a bias is applied and under a high negative-bias are shown in Figs. 5.1(a) and 5.1(b).

5.3 Distribution of the Emitted Current Density

The distribution of electrons incident on the drift-layer/barrier justifiably resembles a drifted heated Maxwellian model. Although the conduction electrons are distributed both in the Γ and L bands, for simplicity it is assumed that the electrons in the GaAs drift layer are all in the Γ -valley. Additionally, the Γ -valley was assumed to be parabolic, leading to a non-dispersive effective mass for electrons at any given wave vector. Here, the diffusion of electrons from the barrier to the drift layer has been neglected, as when the device is close to switching, the diffusion current is negligible compared to the forward current. This is due to the large difference between the fields in the two regions. However, a complete model would take both the electron distribution in the two valleys and take the diffusion current into account.

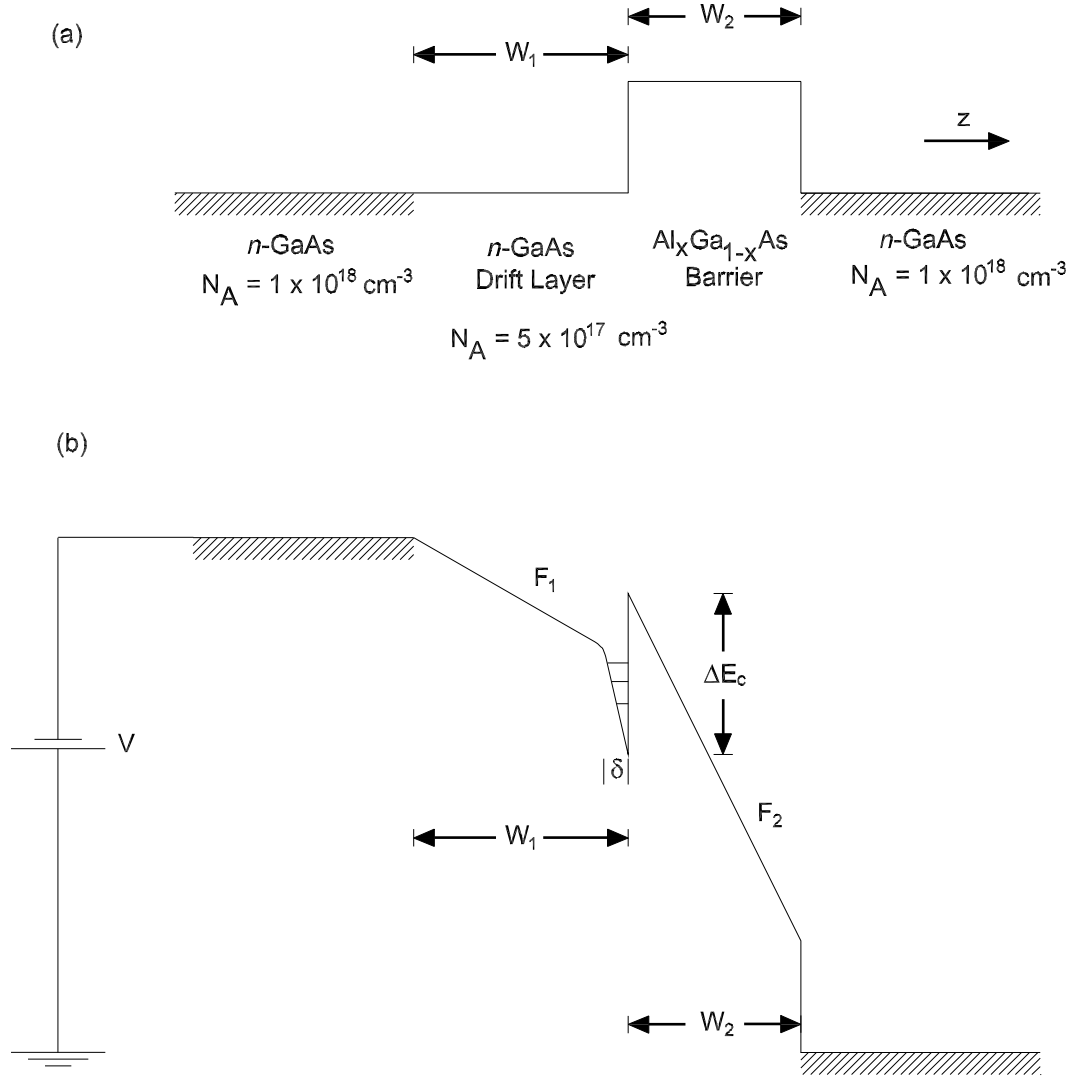


Figure 5.1: Partial band diagram of the GaAs/Al_{0.46}Ga_{0.54}As heterostructure hot-electron device (a) at thermal equilibrium, and (b) under reverse bias. Width of the space charge layer where the carriers trap to form a 2D-electron gas is denoted by δ . The diagram is for high bias where the thermionic emission is the predominant current mechanism. Fields due to external bias are denoted by F_1 and F_2 for drift and barrier layers, respectively.

The current density in the drift layer due to electron drift can be written as:

$$j_{z\Gamma} = en\mu_{\Gamma}F_1, \quad (5.1)$$

where n is the doping density of the drift layer, F_1 is the field in the drift layer, and μ_{Γ} is the mobility of electrons in the Γ -valley.

The current density in the barrier layer is determined by the injection of electrons from the drift layer. The space charge accumulation of electrons at the drift layer/barrier interface is given by:

$$-en_s = \epsilon_2 F_2 - \epsilon_1 F_1, \quad (5.2)$$

where n_s is density of the 2D gas, and ϵ_1 and ϵ_2 are the permittivities of the drift and barrier layers, respectively.

The energy levels of the 2-DEG is obtained using a triangular potential approximation. The three lowest bound states for electrons trapped in the space charge well in the drift layer are given by:

$$\begin{aligned} E_{1\Gamma} &= \left(\frac{\hbar^2}{2m_{\Gamma}} \right)^{1/3} \left(\frac{1.14\pi e^2 n_s}{8\epsilon_1} \right)^{2/3} \\ E_{2\Gamma} &= \left(\frac{\hbar^2}{2m_{\Gamma}} \right)^{1/3} \left(\frac{2.63\pi e^2 n_s}{8\epsilon_1} \right)^{2/3} \\ E_{3\Gamma} &= \left(\frac{\hbar^2}{2m_{\Gamma}} \right)^{1/3} \left(\frac{4.136\pi e^2 n_s}{8\epsilon_1} \right)^{2/3} \end{aligned} \quad (5.3)$$

The injection of the electrons from the contact, in other words the current through the device, is proportional to the distribution of the electrons incident on the barrier interface of the device. Distribution for the Γ -valley electrons is given by:

$$f_{\Gamma}(k_z) = \frac{n_{\Gamma}\hbar}{\sqrt{2\pi m_{\Gamma} k_B T_e}} \cdot \exp(-\hbar^2 |k_z - k_{d\Gamma}|^2 / 2m_{\Gamma} k_B T_e) \quad (5.4)$$

where $\hbar k_{d\Gamma} \equiv \mu_{\Gamma} F_1 m_{\Gamma}$ is the momentum due to drift, $\hbar k_z$ is the quasi-momentum in the z direction, and T_e is the temperature of the electron distribution. The barrier lowering by the space charge accumulation layer is analogous to an increase in the quasi-momentum normal to the interface, $\hbar k_{zn} = \sqrt{2m_{\Gamma} E_{n\Gamma}}$. Hence, the electron distribution in Eq. 5.4 modifies to:

$$f_{\Gamma}(k_z, k_{zn}) = \frac{n_{\Gamma} \hbar}{\sqrt{2\pi m_{\Gamma} k_B T_e}} \cdot \exp(-\hbar^2 |k_z + k_{zn} - k_{d\Gamma}|^2 / 2m_{\Gamma} k_B T_e) \quad (5.5)$$

Electrons drifting towards the $\text{Al}_x\text{Ga}_{1-x}\text{As}$ barrier traverse the narrow space charge layer ballistically, and the shape of the distribution remains constant. This is because the mean free path for inelastic scattering is greater than the width of the space charge layer and the number of carriers that lose energy from such scattering events remains insignificant. Based on this, the tunneling component of the injection current density is given by:

$$j_{T\Gamma} = -e \int_0^{\infty} dk_z f_{\Gamma}(k_z) \frac{\hbar k_z}{m_{\Gamma}} D_{\Gamma}(k_z, F_2) \quad (5.6)$$

where $D_{\Gamma}(k_z, F_2)$ is the tunneling probability for an incident electron with energy $\hbar k_z$ and the barrier electric field. The evaluation of $j_{T\Gamma}$ is done numerically.

5.3.1 Tunneling Probability of the Carriers

The tunneling probability or transmission probability of an electron described by the WKB approximation breaks when the incident energy of the carrier is close to the barrier height at the interface. The following method⁷³ well describes the tunneling probability for any incident energy of the carrier. The transmission coefficient, $D_{\Gamma}(k_z, F_2)$, and the tunneling current densities, $j_{T\Gamma}$ and j_{TB} , for both trapezoidal and triangular barriers are presented as a function of the voltage applied across the device (see Barrier in appendix for details).

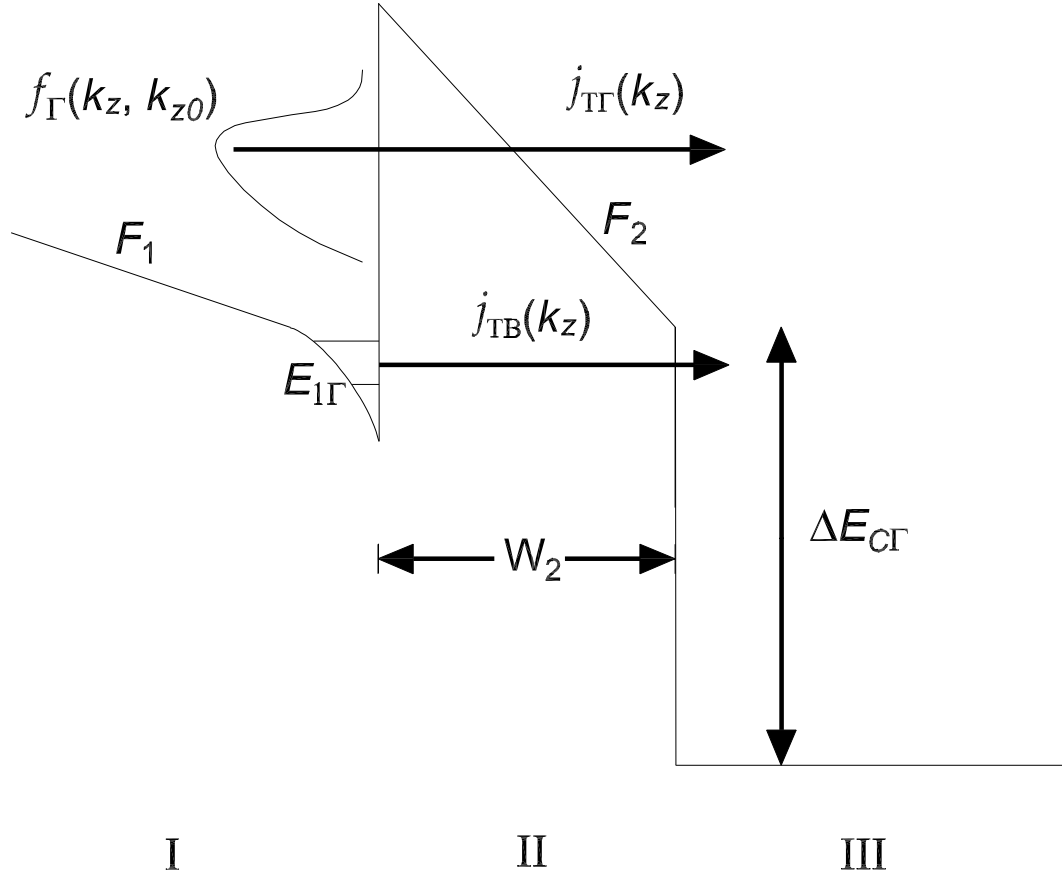


Figure 5.2: Conduction band edge of HHED under bias. Subscript Γ refers to the Γ -valley of the conduction band. The tunneling current components from the bulk electrons in the drift layer ($j_{T\Gamma}(k_z)$) and the bound electrons in the V-shaped well in the space charge region ($j_{TB}(k_z)$) are shown by arrows. The band bending in the barrier due to space charge is negligible and the field in the barrier is assumed to be constant. The label Γ represents the electrons in that valley.

$$\begin{aligned}
D_{\Gamma}(k_z, F_2) = & \frac{4k_{\text{III}}}{\pi^2 k_{\text{I}}} \left(\left\{ [\text{Ai}(z_{\text{I}})\text{Bi}'(z_{\text{III}}) - \text{Bi}(z_{\text{I}})\text{Ai}'(z_{\text{III}})] \right. \right. \\
& + \frac{k_{\text{III}}}{k_{\text{I}}} [\text{Bi}'(z_{\text{I}})\text{Ai}(z_{\text{III}}) - \text{Ai}'(z_{\text{I}})\text{Bi}(z_{\text{III}})] \left. \right\}^2 \\
& + \left\{ \frac{\lambda}{k_{\text{I}}} [\text{Ai}'(z_{\text{I}})\text{Bi}'(z_{\text{III}}) - \text{Bi}'(z_{\text{I}})\text{Ai}'(z_{\text{III}})] \right. \\
& \left. \left. + \frac{k_{\text{III}}}{\lambda} [\text{Ai}(z_{\text{I}})\text{Bi}(z_{\text{III}}) - \text{Bi}(z_{\text{I}})\text{Ai}(z_{\text{III}})] \right\}^2 \right)^{-1} \quad (5.7)
\end{aligned}$$

where $z_{\text{I}} = \left(\frac{2m_{\Gamma}}{e^2 F_2^2 \hbar^2} \right)^{1/3} \cdot (\Delta E_{\text{CT}} - E_z)$, $z_{\text{III}} = \left(\frac{2m_{\Gamma}}{e^2 F_2^2 \hbar^2} \right)^{1/3} \cdot (\Delta E_{\text{CT}} - E_z - eF_2 W_2)$, and $\lambda = - \left(\frac{2m_{\Gamma} e F_2}{\hbar^2} \right)^{1/3}$. Here, m_{Γ} is the effective mass of the electrons, F_2 is the field in the barrier, E_z is the z -component of energy of the electrons incident on the barrier, Ai and Bi are the Airy functions. Here, $k_{\text{I}} = \frac{\sqrt{2m_{\Gamma} E_z}}{\hbar}$ and $k_{\text{III}} = \frac{\sqrt{2m_{\Gamma} (E_z + eF_2 W_2)}}{\hbar}$

For an $\text{Al}_x\text{Ga}_{1-x}\text{As}$ with high x fraction, the effective masses in the drift layer and the barrier layer will be different. The derivation has been modified to include the mass difference in the drift layer and the barrier. The modified transmission that considers the mass ratio is given by:

$$\begin{aligned}
D_{\Gamma}(k_z, F_2) = & \frac{4k_{\text{III}}}{\pi^2 k_{\text{I}}} \left(\left\{ [\text{Ai}(z_{\text{I}})\text{Bi}'(z_{\text{III}}) - \text{Bi}(z_{\text{I}})\text{Ai}'(z_{\text{III}})] \right. \right. \\
& + \frac{k_{\text{III}}}{k_{\text{I}}} [\text{Bi}'(z_{\text{I}})\text{Ai}(z_{\text{III}}) - \text{Ai}'(z_{\text{I}})\text{Bi}(z_{\text{III}})] \left. \right\}^2 \\
& + \left\{ \frac{\lambda m_{\text{I}\Gamma}}{k_{\text{I}} m_{\text{III}\Gamma}} [\text{Ai}'(z_{\text{I}})\text{Bi}'(z_{\text{III}}) - \text{Bi}'(z_{\text{I}})\text{Ai}'(z_{\text{III}})] \right. \\
& \left. \left. + \frac{k_{\text{III}} m_{\text{III}\Gamma}}{\lambda m_{\text{I}}} [\text{Ai}(z_{\text{I}})\text{Bi}(z_{\text{III}}) - \text{Bi}(z_{\text{I}})\text{Ai}(z_{\text{III}})] \right\}^2 \right)^{-1} \quad (5.8)
\end{aligned}$$

where $m_{\text{I}\Gamma}$ and $m_{\text{III}\Gamma}$ are the electron effective mass in the drift layer and the barrier, respectively.

As the bias is increased the transmission coefficient, hence the tunneling current, in the device increase. The field F_1 grows linearly with the forward current provided that the diffusion current is negligible. Starting with a particular bias, the electrons in the drift layer get heated by the injection from the contact, leading to thermionic emission. Once the thermionic current begins to flow the current becomes unstable and switches rapidly to the high conductance regime with a low series resistance.

By considering only the lowest bound state of the space charge well, the tunneling current from the 2-DEG is given by:

$$j_{TB} = -en_s \frac{E_{1\Gamma}}{\hbar} D_{\Gamma}(E_{1\Gamma}, F_2). \quad (5.9)$$

The total current density in the device is then given by:

$$j_T = j_{TT} + j_{TB}. \quad (5.10)$$

By setting the input power density in the drift layer region to the average energy loss rate to polar optical phonons in the Γ -valley the electron temperature in the drift layer can be calculated. Here, we assume that the space charge accumulation layer does not change or has a negligible effect on the electron temperature. An approximate expression for the energy loss rate of electrons to polar optical phonons in the Γ -valley is given by:

$$j_{z\Gamma} F_1 = en \left(\frac{2k_B \theta_D}{\pi m_{\Gamma}} \right)^{1/2} \cdot F_0 \frac{e^{x-x_e} - 1}{e^x - 1} x_e^{1/2} e^{x_e/2} K_0 \left(\frac{x_e}{2} \right), \quad (5.11)$$

where θ_D is the Debye temperature, and x and x_e are energy of optical phonons in terms of the energy corresponding to device temperature and the electron temperature, respectively; so $x = \hbar\omega_{LO}/k_B T$ and $x_e = \hbar\omega_{LO}/k_B T_e$. F_0 in the above equation is an effective

electric field which determines the strength of the electron coupling to the polar modes⁷⁴.

Specifically,

$$eF_0 \equiv \left(\frac{me^2}{\hbar^2} \right) \hbar\omega_{LO} \left(\frac{1}{\epsilon_\infty} - \frac{1}{\epsilon_0} \right) \quad (5.12)$$

where $\hbar\omega_{LO}$ is the longitudinal optical phonon energy, and ϵ_∞ and ϵ_0 are the high frequency and static dielectric constants of the drift layer, respectively. For GaAs material, $F_0 = 5.95$ kV/cm.

The bias voltage of the device is related to the fields in the drift and barrier layers by:

$$-V = W_1 F_1 + W_2 F_2. \quad (5.13)$$

Here, the bias drop in the space charge layer has been neglected due to its width δ being negligible compared to the layer thickness.

The magnitude of $E_{n\Gamma}$'s can be found using equations Eqs. 5.1, 5.2, 5.3, and 5.13. The bias voltage where the HHED switches from the low to high conductance branch was determined iteratively. The iteration was initialized by setting the temperature of the electron population at the device temperature, and the drift momentum, k_z to zero. Then, Eqs. 5.11 and Eqs. 5.6 were solved iteratively until the current in the drift layer $j_{z\Gamma}$ converges. As we increase the bias voltage by small steps, at one particular voltage $j_{z\Gamma}$ fails to converge, thus deriving the switching point.

The above model was implemented, using the described numerical procedure, on a hypothetical HHED structure consisting of a 90 nm-thick n -type GaAs drift layer and a 500 nm-thick $\text{Al}_{0.45}\text{Ga}_{0.55}\text{As}$ barrier layer. The drift layer doping density was selected as $5 \times 10^{17} \text{ cm}^{-3}$. The variation of the electron temperature in the drift layer was studied

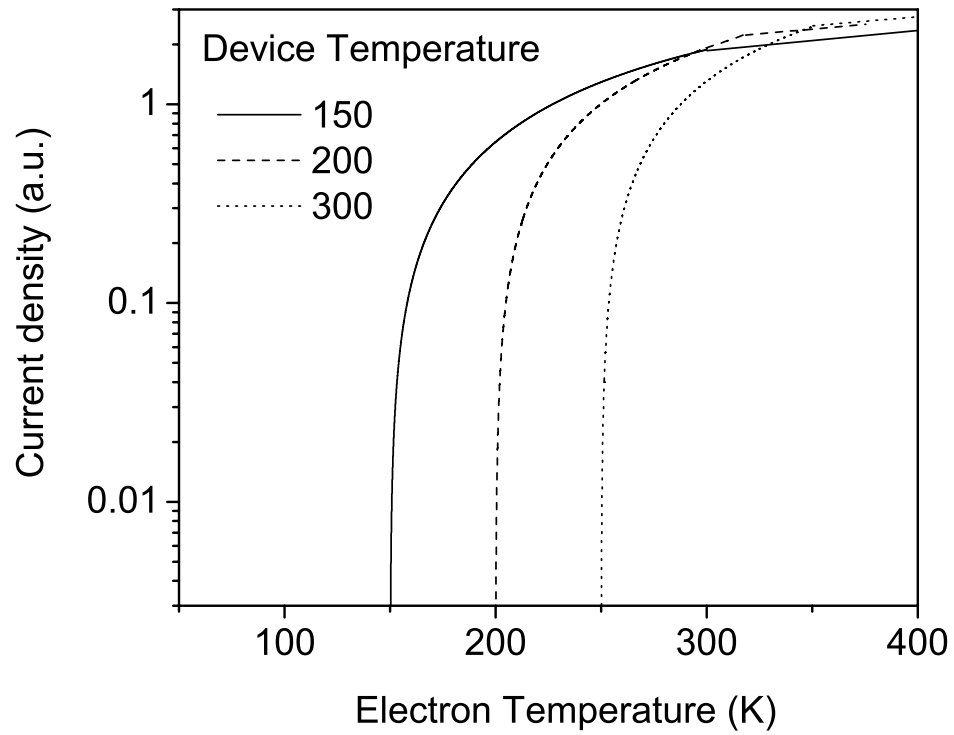


Figure 5.3: Current density vs. Electron temperature in the drift layer of an HHED. The electron temperature increases rapidly after a certain current signaling the switching from the low conductance to the high conductance branch.

for different device temperatures. Before the iteration was started, the thermal equilibrium condition of the electron population was insured by fixing the electron temperature to be the same as the device temperature. The results for three different lattice temperatures are shown in Fig. 5.3. The electron temperature grows almost linearly with the current and at a particular current value (or bias value) it starts increasing in an unbounded fashion. Each data point shown for a given curve is a converging solution of the iteration discussed earlier, and the last point of the curve is where the iteration becomes unstable and grows unbounded. This is the point where the device starts switching from the low conductance to high conductance branch. After this point the electron temperature should increase indefinitely only to be limited by an upper limit of ~ 3000 K, which corresponds the electron transfer energy (~ 250 meV) from the Γ to the L-valley for GaAs.

The energy distribution of the emitted current density corresponding to Γ -valley electrons are shown in Fig. 5.4. This shows the dependence of the logarithm of the emitted current for two different applied fields at different device temperatures. At these bias fields, the distribution shows that the current deviates mainly from the thermal-assisted-field-emission dominated to the thermal emission dominated mechanism as the device temperature increases from 50 K to 300 K. However, as the electrons in the drift layer get heated by the injection current from the contact, the distribution for any given temperature becomes unstable and drifts above the barrier, switching the device into the high conductance region. Even at 300 K, the total current is not due to thermal emission completely, and the increase in the barrier lowering with the applied bias moves the barrier (shown by the vertical dotted line) down. Here, this has been implemented by shifting the distribution up along the

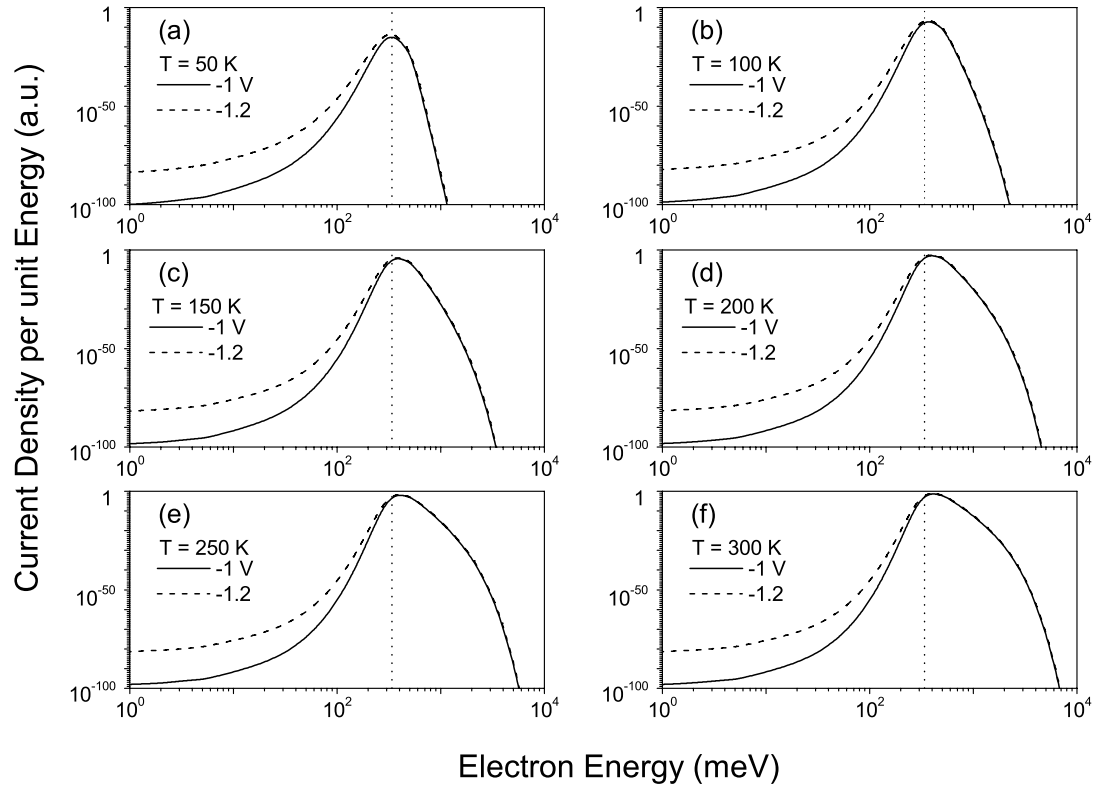


Figure 5.4: Energy distribution of the emitted current density, corresponding to Γ -valley electrons, for six different values of the temperature. The solid and dashed lines correspond to bias values of 1.0 and 1.2 V, respectively. The dotted line shows the position of the barrier. As the temperature increases the distribution shifts above the interfacial barrier.

energy scale to match the corresponding barrier lowering, while keeping the barrier fixed. The barrier lowering, through increasing the bias, will drive the device towards the unstable point for switching. Therefore, one would expect the device to switch at a lower voltage with increasing lattice temperature.

The current density variation at different temperatures is shown in Fig. 5.5. Each data point in a given curve shows the convergence of the current starting from a zero drift velocity for a given bias. The number of iterations for a single bias is fixed at a maximum of 1000, and the current density at most bias points converged within ~ 200 iterations. The point where the discontinuity of the convergence (in other words the last point of the plot) shown is where the electron temperature increases to rise rapidly and the current did not converge even for 1000 iterations. As expected, the switching bias drops with increasing temperature, and this was illustrated with electron emission in the previous paragraph. In a practical device under current bias configuration, this would show up as a jump between two voltage points.

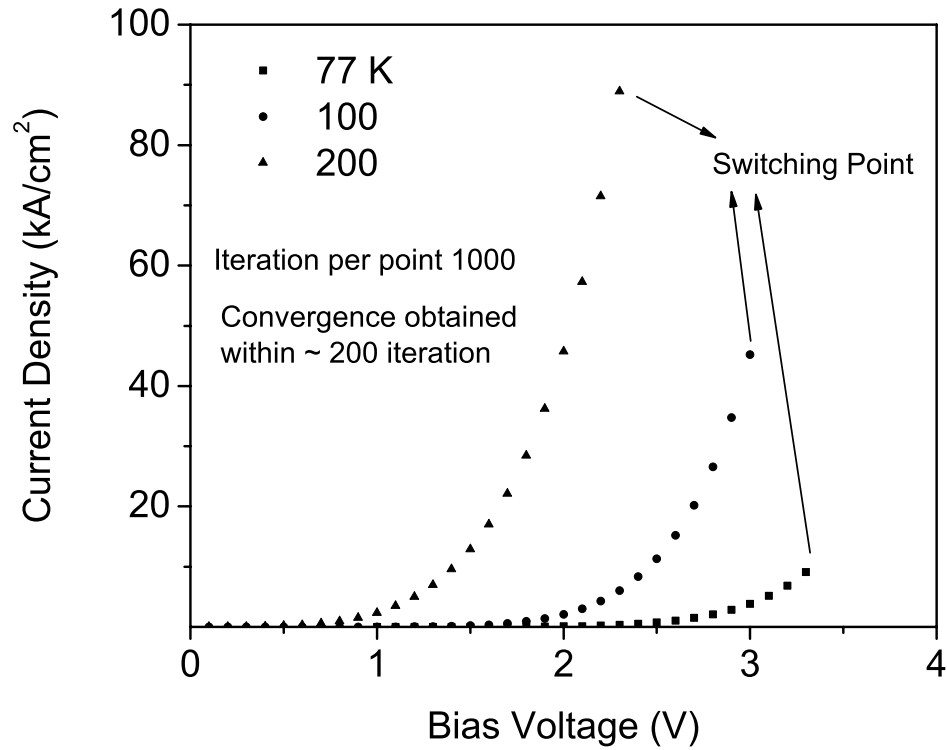


Figure 5.5: Variation of the current density with lattice temperature. The label “switching point” indicates the bias voltage where the device switches from a low conductance to a high conductance branch. The switching bias decreases with the temperature as shown. This is the result of the shifting of Γ -valley electron distribution from the field-emission dominated to thermal-emission dominated regime with the increasing temperature (see Fig. 5.4), reducing the field required to drive the device to switching point.

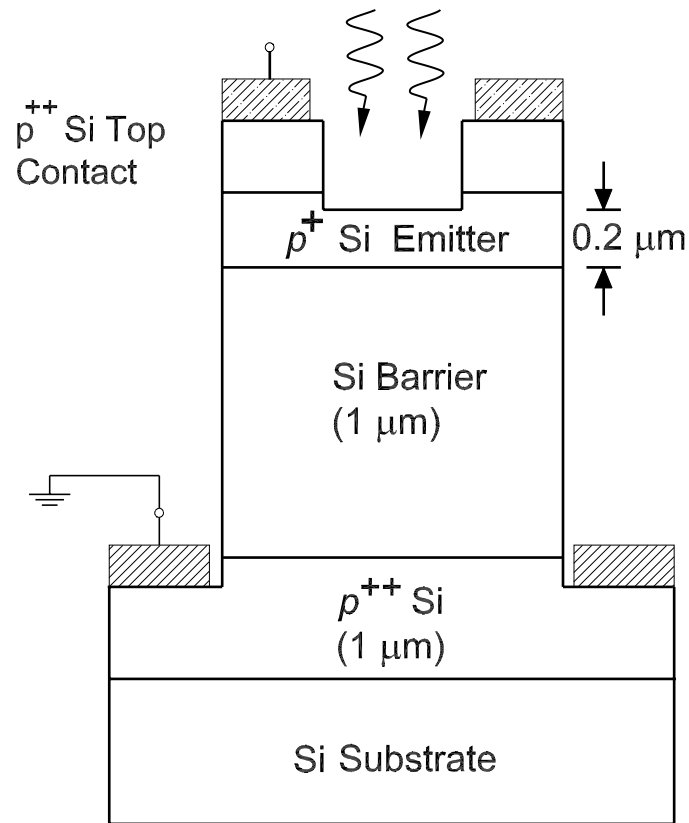


Figure 5.6: Schematic diagram of the Si HIWIP device that shows switching behavior. The doping density of the emitter region is $2.5 \times 10^{18} \text{ cm}^{-3}$. Although intentionally undoped, the Si barrier shows some doping migration as photoionization from coulomb wells were observed.

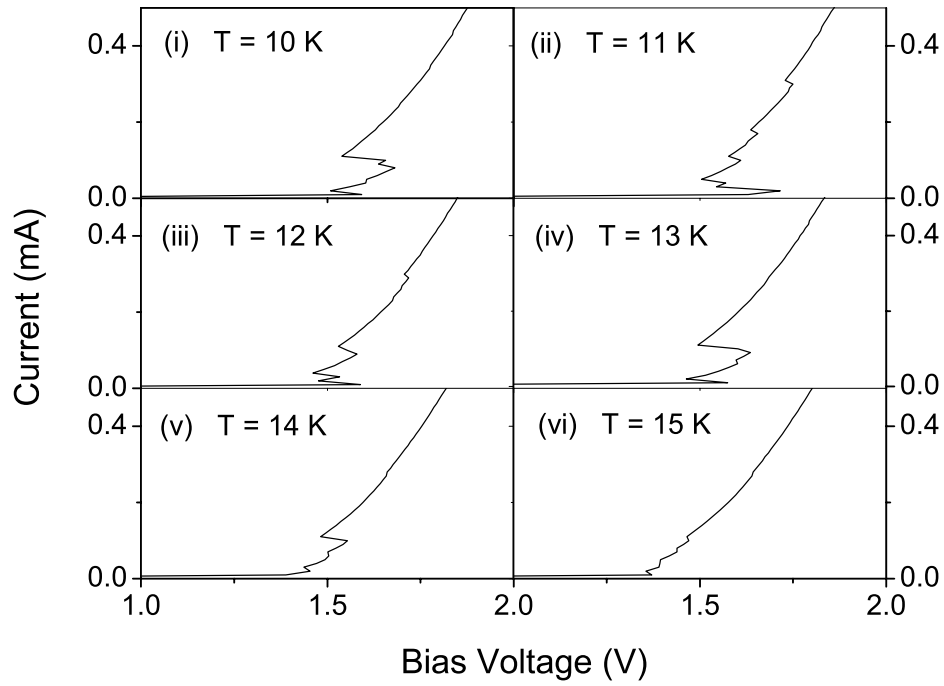


Figure 5.7: Current voltage curves for the single barrier *p*-type Si-homojunction device at different temperatures. These were obtained under current bias mode. The sharp jumps in the curves were observed as the field across the 1 μm -thick barrier reaches ~ 15 kV/cm. The switch to the high conductance branch is most likely due to the tunneling of carriers at Fermi level through the barrier. As the temperature was increased, the field required for this reduced slightly as expected, and was described before (see current density spectra in Fig. 5.4).

5.3.2 Switching Observed in a Si-Homojunction Device

A switching type current-voltage characteristics were observed in a C-doped Si-HIWIP device. This device was originally designed for FIR detection with a threshold frequency of $40\ \mu\text{m}$. The structure of the device is shown in Fig. 5.6. During the dark current measurements, several sharp jumps were noticed at high fields, especially at low temperatures. Subsequently, as expected, when the device was biased with a current source, several small sharp turns were observed at low temperatures as shown in Fig. 5.7. These jumps may be due to the tunneling of the carriers in the Fermi level when the barrier field increases up to $\sim 15\ \text{kV/cm}$. At these fields the barrier becomes triangular, presenting a small tunneling width to the incoming carrier, thereby increasing the emission.

In order to confirm that the current-voltage characteristic does not change with any space charge effect in the device the time delay between the measurements was increased to 0.5 s and the number of data points averaged was increased to 100. The three main jumps were still observed without any change. Although the detector shows several prominent sharp turn-on features in its current-voltage measurements, pulsing from this was not observed under both voltage and current bias configurations. This is because IV does not have a true S-shaped NDR region.

5.4 Conclusion

The simple model described above is based on the field heating of 2-D carrier population trapped at the interface. The involvement of the materials is through many of their electrical and optical constants such as the mobility and phonon energy for examples.

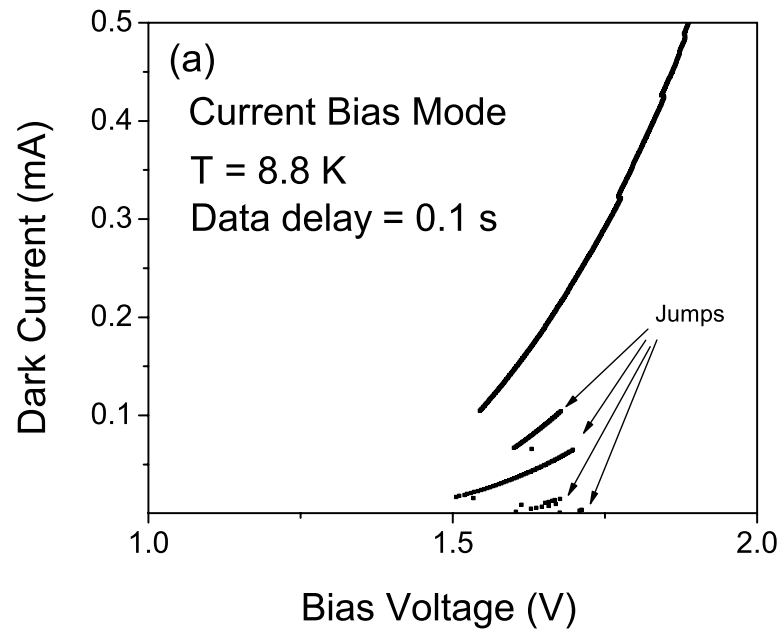


Figure 5.8: Current voltage curves for the single barrier *p*-type Si-homojunction detector with 100 data point average and a data delay of 0.1 s. This was done at different temperatures obtained with the current bias mode. The sudden jump in the curve occurs when the field across the 1 μm -thick barrier reaches $\sim 15 \text{ kV/cm}$. The switch to the high conductance ranch is most likely due to the carbon impurities in the barrier. The gap reduces with increasing temperature.

A high field is required to switch the 2D-electron population from its lattice equilibrium temperature to a 3D-hot electron population, and this depends both on the material parameters of the HHED and the barrier height. The room temperature operation of HHED is obtained using a barrier ~ 20 KT . The reality of the pulsing in experimental devices would most likely be the impact-ionization. By modeling the impact ionization, which is always strenuous, a better control of the pulsing device can be achieved. Besides, current filamentation patterns, which are typical of high electric field devices, such as the device discussed in this and IMPATT diodes, should be monitored at real time to understand the anomalies of current density existing between the present model and experimental results. As the materials used in this design are GaAs and AlGaAs, the hot spots (local variations in the temperature) produced by current channeling can be identified from infrared emission from the device.

Appendix A

Programs Used for Response Modeling

The following program is written in C++ (version 6.0). The program is a collection of classes that includes a “Dielectric stack” and “Infrared detector” classes. These allow the calculation of reflectance, transmittance, and absorptance of dielectric stacks for both ‘s’ and ‘p’ polarized, and unpolarized light for a given input angle. A normalized refractive index method has been used throughout. The class “Ambient” will allow independent modification of the permittivity and the permeability of the front and back surroundings of the detector. Variation of the responsivity can be plotted as a function of both the wavenumber and incident angle. The units used throughout is “cm” for lengths and “cm⁻¹” for wavenumber to avoid complications of transformation from one unit to the other. The permittivity function call in “Dielectric layer” class evaluates the contribution to permittivity from both the free carrier density in a layer and the phonons of the layer material. By modifying the

permittivity mechanism to that of interband transition, the responsivity in this region can be evaluated. The variables needed for this can be either added in to corresponding material files in the optical library or can be input by the user. An added advantage is the ability to calculate the device response for both the “front illumination” and “back illumination” configurations. Here, it should be noted that for both configurations “Top contact” is the very top layer on the substrate. However, for “bottom illumination” the legends in the responsivity plot and the corresponding data column reads top and bottom in the direction of light travel.

A.1 Class Definitions and Their Implementations

The following are the main classes used in the program. The “infraredDetector” class is derived from “DielectricStack” class, and the “BoundOscillator” class is derived from the class “FreeOscillator”. Overloaded functions and operator overloads are included in many of these classes.

- FresnelMatrixCoefficient
- ThinFilmTransferMatrix
- FreeOscillator
- BoundOscillator
- Carrier
- DielectricLayer
- Ambient
- DielectrucStack
- InfraredDetector

```

//Copyrights - M. B. Rinzan, Optoelectronics Lab of Georgia State University

//Class definition for Oscillator Definition

#ifndef OSCILLATOR_H
#define OSCILLATOR_H

#include "OpticalConstants.h"

class OpticalConstants;
using namespace std;

class FreeOscillator
{
protected:
    double ResonanceFrequency;
    double Damping;

    FreeOscillator (double Resonance_Frequency =0., double _Damping =0.)
        :ResonanceFrequency(Resonance_Frequency), Damping(_Damping){}

public:
    void SetResonanceFrequency(const double& Resonance_Frequency = 0.);
    void SetDamping          (const double& _Damping = 0.);
    virtual void SetStrength  (const double& _Strength = 0.){};

    double GetResonanceFrequency(void) {return ResonanceFrequency;}
    double GetDamping(void) {return Damping;}
    FreeOscillator GetOscillator(void) {return *this;};

    virtual inline void Erase(void)
    {
        FreeOscillator Oscillator;
        *this = Oscillator;
    }

    inline friend istream& operator>> (istream& in, FreeOscillator& Free_Oscillator)
    {
        cout<<"\n\tEnter Resonance Frequency (cm^-1): ";
        in>>Free_Oscillator.ResonanceFrequency;

        cout<<"\n\tEnter Broadening (cm^-1): ";
        in>>Free_Oscillator.Damping;

        return in;
    }
}

```

```

inline friend ostream& operator<<(ostream& out, const FreeOscillator& Free_Oscillator)
{
    if (Free_Oscillator.ResonanceFrequency != 0.) {
        out <<Free_Oscillator.ResonanceFrequency<<" ";
        out<<Free_Oscillator.Damping;
    }
    return out;
}

/*This is allowed so that Oscillators can be constructed/Accessed only within
  DielectricLayer Objects, which are meaningless otherwise*/

friend class DielectricLayer;

friend void OpticalConstants::SetNonDispersiveConstantsForLayer
    (DielectricLayer& DLayer, int LayerIndex, int InputOption);

friend void OpticalConstants::OpticalConstantsFromUser
    (DielectricLayer& DLayer);

friend void OpticalConstants::ShowOpticalConstants
    (ostream& out, const DielectricLayer& DLayer);
};

class BoundOscillator:public FreeOscillator
{
private:
    double Strength;

public:
    BoundOscillator (double Resonance_Frequency = 0., double _Damping = 0.,
                    double _Strength = 0.)
        :FreeOscillator(Resonance_Frequency, _Damping), Strength(_Strength){}

    double GetStrength(void) {return Strength;}
    BoundOscillator GetOscillator(void) {return *this;};

    inline void Erase(void)
    {
        BoundOscillator Oscillator(0., 0., 0.);
        *this = Oscillator;
    }

    void SetStrength (const double& _Strength = 0.);

```

```

inline friend istream& operator>> (istream& in, BoundOscillator& Bound_Oscillator)
{
    cout    <<"\n\tEnter Resonance Frequency (cm^-1): ";
    in      >>Bound_Oscillator.ResonanceFrequency;

    cout    <<"\n\tEnter Broadening (cm^-1): ";
    in      >>Bound_Oscillator.Damping;

    cout    <<"\n\tEnter Strength: ";
    in      >>Bound_Oscillator.Strength;

    return in;
}

inline friend ostream& operator<< (ostream& out,
                                   const BoundOscillator& Bound_Oscillator)
{
    if (Bound_Oscillator.ResonanceFrequency != 0.) {
        out    <<Bound_Oscillator.ResonanceFrequency<<" "
               <<Bound_Oscillator.Damping<<" "
               <<Bound_Oscillator.Strength;
    }
    return out;
}

friend class DielectricLayer;

friend void OpticalConstants::SetNonDispersiveConstantsForLayer
(DielectricLayer& DLayer, int LayerIndex, int InputOption);

friend void OpticalConstants::OpticalConstantsFromUser
(DielectricLayer& DLayer);

friend void OpticalConstants::ShowOpticalConstants
(ostream& out, const DielectricLayer& DLayer);

};

#endif

```

```
//Implementation of the class Oscillator
#include "stdafx.h"
#include "Oscillator.h"

void FreeOscillator::SetResonanceFrequency(const double& Resonance_Frequency)
{
    ResonanceFrequency = Resonance_Frequency;
}

void FreeOscillator::SetDamping(const double& _Damping)
{
    Damping = _Damping;
}

void BoundOscillator::SetStrength(const double& _Strength)
{
    Strength = _Strength;
}
```

```

//Declaration for Class Carrier
#ifndef CARRIER_H
#define CARRIER_H

#include <iostream>
#include "Constants.h"
#include "WaveVector.h"

using namespace std;

class WaveVector;

class Carrier
{
    static double Charge;
    static double FreeMass;

private:
    char          Type;
    double        Density;
    double        EffectiveMass;
    double        Mobility;
    double        ScatteringTime;
    double        Temperature;
    WaveVector    RandomWaveVector;
    WaveVector    DriftWaveVector;

public:
    Carrier (void):Type("?"), Density(0.){};
    Carrier (char _Type, double _Density, double Effective_Mass = 0.,
              double _Mobility = 0., double Scattering_Time = 0., double _Temperature = 0.);
    ~Carrier(){ };

    void SetEffectiveMass(const double& Effective_Mass);
    char  GetType          (void) const {return Type;}
    double GetMobility      (void) const {return Mobility;}
    double GetDensity       (void) const {return Density;}
    double GetEffectiveMass (void) const {return EffectiveMass;}
    void   SetDriftWaveVector (WaveVector Wave) {DriftWaveVector = Wave;}
    void   SetRandomWaveVector (WaveVector Wave)
            {RandomWaveVector = Wave;}

    WaveVector GetDriftWaveVector (void) const {return DriftWaveVector;};
    WaveVector GetRandomWaveVector (void) const {return RandomWaveVector;};
    void ListContent (void);
    void Erase (void);

```



```

inline bool operator==(const Carrier& _Carrier) const
{
    return (_Carrier.Type == Type && _Carrier.Density == Density );
}

inline bool operator!=(const Carrier& _Carrier) const
{
    return !(operator==( _Carrier));
}

inline friend ostream& operator<<(ostream& out, const Carrier& _Carrier)
{
    return out <<_Carrier.Type<<"\t"
               <<_Carrier.Density<<" (cm^-3)";
}

inline friend istream& operator>>(istream& in, Carrier& _Carrier)
{
    in>>_Carrier.Type>>_Carrier.Density;
    _Carrier.Type = tolower(_Carrier.Type);

    return in ;
}

//Carriers can be accessed only within DielectricLayer Objects
friend class DielectricLayer;

};

#endif

```

```

//Implementation of class Carrier

#include "stdafx.h"
#include "Carrier.h"

double Carrier::Charge    = 1.602e-19;           //C
double Carrier::FreeMass  = 9.10938188e-31;      //kg

Carrier::
Carrier(char _Type, double _Density, double Effective_Mass, double _Mobility,
         double Scattering_Time, double _Temperature)
    :Type(_Type),
     Density(_Density),
     EffectiveMass(Effective_Mass),
     Mobility(_Mobility),
     ScatteringTime(Scattering_Time),
     Temperature(_Temperature),
     DriftWaveVector(),
     RandomWaveVector() {}

void Carrier::Erase(void)
{
    Type = '?';
    Density = 0.;
    EffectiveMass = 0.;
    Mobility = 0.;
    ScatteringTime = 0.;
    Temperature = 0.;
    DriftWaveVector.Erase();
    RandomWaveVector.Erase();
}

void Carrier::ListContent(void)
{
    cout << *this << endl
         << "Effective Mass:\t" << EffectiveMass << endl
         << "Temperature:\t" << Temperature << endl
         << "Mobility:\t" << Mobility << endl
         << "Scattering Time:\t" << ScatteringTime << endl
         << "Carrier Drift(cm^-1):\n" << DriftWaveVector << endl;
}

void Carrier::SetEffectiveMass(const double &Effective_Mass)
{
    EffectiveMass = Effective_Mass;
}

```

```

//Const Definitions
#ifndef CONSTANTS_H
#define CONSTANTS_H

#define pi 3.141593
#define e 1.602189e-19
#define mo 9.109534e-31
#define h 6.62620e-34
#define hBar 1.05459e-34
#define BK 1.380662e-23
#define pmt 8.85418782e-12
#define pmb 1.256637E-6
#define c pow(pmt*pmb, -0.5)
#define n_Vacuum 1.0

#define GraphSleepLength 1000

#endif

//header file for utilities
#ifndef Read_H
#define Read_H

#include <stdlib.h>
#include <iostream>
#include <string>
#include <fstream>
#include <conio.h>
#include "Carrier.h"

using namespace std;

string  String (int x, const string& prompt);
char    Char (const string& prompt);
double  Double (const string& prompt);
int     Int (const string& prompt);

template <class T>
T Message (int x, const string& prompt);

void     OutText (const string& str, bool bLineFeed = false);
string   AskUser (const string& prompt, bool bMakeUpper, bool bMandatory);
void     OpenFile (std::ifstream& Infile, string FileName = "?");

#endif

```

```

#include "stdafx.h"
#include "Read.h"

using namespace std;

void OutText (const string& str, bool bLineFeed)
{
    cout << str;
    if (bLineFeed)
        cout << endl;
}

string AskUser (const string& prompt, bool bMakeUpper = false, bool bMandatory = true)
{
    OutText(prompt);
    string input;
    while (true) {
        getline(cin, input);
        if (!bMandatory || !input.empty())
            break;
    }

    if (bMakeUpper)
        for (int i = 0; i < input.length(); ++i)
            input[i] = toupper(input[i]);

    return input;
}

void OpenFile (std::ifstream& Infile, string FileName)
{
    if (FileName == "?") {
        do {
            Infile.clear();
            FileName = AskUser("File name?", false, true);
            Infile.open(FileName.c_str(), ios::in);
            if (!Infile.is_open())
                cout << "Error! " << FileName << " does not exist\n";
        } while (!Infile.is_open());
    }
    else {
        Infile.clear();
        Infile.open(FileName.c_str(), ios::in);
        if (!Infile.is_open()) {
            cout << "Error! " << FileName << " does not exist\n";
            exit(1);
        }
    }
}

```

```

//Definition of Class FresnelCoefficient Matrix

#ifndef FRESNEL_COEFFICIENT_MATRIX
#define FRESNEL_COEFFICIENT_MATRIX

#include <iostream>
#include <math.h>
#include <complex>
#include "Constants.h"
#include "Wavevector.h"

using namespace std;

struct Wave {
    COMPLEX Electrical;
    COMPLEX Magnetic;
};

struct Intensity {
    double Magnitude;
    double PhaseDifference;
};

class FresnelCoefficientMatrix
{
private:
    Wave      rs;
    Wave      rp;
    Wave      ts;
    Wave      tp;
    Intensity Rs;
    Intensity Rp;
    Intensity Ts;
    Intensity Tp;

public:

    friend class ThinFilmTransferMatrix;
    friend class Ambient;
    friend class DielectricLayer;
    friend class DielectricStack;

    friend std::ostream& operator<< (ostream& Out, const Wave& _Wave)
    {
        double PhaseDifference_Electrical;
        double PhaseDifference_Magnetic;

        PhaseDifference_Electrical = atan(imag(_Wave.Electrical)/real(_Wave.Electrical));
        PhaseDifference_Magnetic = atan(imag(_Wave.Magnetic)/real(_Wave.Magnetic));
    }
};

```

```

        if (PhaseDifference_Electrical < 0 )
            PhaseDifference_Electrical += pi;

        if (PhaseDifference_Magnetic < 0 )
            PhaseDifference_Magnetic += pi;

        return Out          <<sqrt(norm(_Wave.Electrical))
                           <<"\t"<<PhaseDifference_Electrical
                           <<"\t"<<sqrt(norm(_Wave.Magnetic))
                           <<"\t"<<PhaseDifference_Magnetic;
    }

    friend std::ostream& operator<< (ostream& Out, const Intensity& _Intensity)
    {
        return Out          <<_Intensity.Magnitude;
                           <<"\t"<<_Intensity.Phase;
    }

    friend ThinFilmTransferMatrix TransferMatrix
        (DielectricLayer& InLayer, DielectricLayer& OutLayer, Ambient& IncidentMedium,
         const double WaveNummber, const double IncidentAngle, const char& Polarization);

    friend ThinFilmTransferMatrix TransferMatrix
        (Ambient& IncidentMedium, DielectricLayer& OutLayer, const double WaveNummber,
         const double IncidentAngle, const char& Polarization);

    friend ThinFilmTransferMatrix TransferMatrix
        (DielectricLayer& InLayer, Ambient& EmergingMedium, Ambient& IncidentMedium,
         const double WaveNummber, const double IncidentAngle, const char& Polarization);

};

#endif

```

```
//Declaration for class OpticalParameters
```

```
#ifndef OpticalConstants_H
#define OpticalConstants_H
```

```
#include<iostream>
#include<math.h>
#include<complex>
#include<vector>
#include "Read.h"
```

```
using namespace std;
```

```
class DielectricLayer;
class DielectricStack;
```

```
class OpticalConstants
```

```
{
```

```
    private:
```

```
        string    DefaultPath;
        void OpticalConstant(){ };
```

```
    public:
```

```
        void SetDefaultPath (const string& DPath) { DefaultPath = DPath; };
        void SetNonDispersiveConstantsForStack (DielectricStack& DStack);
        string GetDefaultPath (void) {return DefaultPath;}
        void SetNonDispersiveConstantsForLayer
            (DielectricLayer& DLayer, int InputOption, int LayerIndex);
```

```
        void OpticalConstantsFromUser (DielectricLayer& DLayer);
        void ShowOpticalConstants (ostream& out, const DielectricLayer& DLayer);
```

```
};
```

```
#endif
```

```

//Implementation of OpticalConstants Header with peripheral Functions for evaluation
//The Functions implements an Equation Parser

#include "stdafx.h"
#include "DielectricLayer.h"
#include "DielectricStack.h"
#include "OpticalConstants.h"
#include "Read.h"
#include <sstream>

using namespace std;

void Trim(char* ); // Function to eliminate blanks
double Evaluate(char* ); // Function evaluating an expression
double Term(char*, int& ); // Function analyzing a term
double Number(char*, int& ); // Function to recognize a number
char* Extract(char*, int& ); // Function to extract a substring
double Parse (string , const double , const double );

const int ExpressionLength = 100; // Maximum expression length

// Function to eliminate blanks from a string
void Trim(char* str)
{
    int i=0; // 'Copy to' index to string
    int j=0; // 'Copy from' index to string

    while((* (str+i) = *(str+j++)) != '\0')
        if(* (str+i) != ' ')
            i++;
}

// Function to evaluate an arithmetic expression
double Evaluate(char* str)
{
    double value = 0;
    int index = 0; // Keeps track of current character position

    value = Term(str, index);
    for(;;) {
        switch(* (str+index++)) {
            case '\0': return value;
            case '+': value += Term(str, index); break;
            case '-': value -= Term(str, index); break;
            default: cout<<"\nExpression error"<<endl;
                    exit(1);
        }
    }
}

```


// Function to get the value of a term

```
double Term(char* str, int& index)
{
    double value = 0;
    value = Number(str, index);

    while((*str+index)=='*') {
        if((*str+index)=='*') value *= Number(str, ++index);
        if((*str+index)=='/') value /= Number(str, ++index);
    }

    return value;
}
```

// Function to recognize an expression in parentheses or a number in a string

```
double Number(char* str, int& index)
{
    double value = 0.0;
    if((*str+index) == '(') {
        char* psubstr = 0; // Pointer for substring
        psubstr = Extract(str, ++index); // Extract substring in brackets
        value = Evaluate(psubstr); // Get the value of the substring
        delete[] psubstr; // Clean up the free store

        return value;
    }

    while(isdigit(*str+index))
        value = 10*value + (*str+index++) - 48;

    if((*str+index) != '.')
        return value;

    double factor = 1.0; // Factor for decimal places

    while(isdigit(*str(++index))) {
        factor *= 0.1; // Decrease factor by factor of 10
        value = value + (*str+index-48)*factor; // Add decimal place
    }

    return value; // On loop exit we are done
}
```

```

// Function to extract a substring between parentheses (requires string)
char* Extract(char* str, int& index)
{
    char buffer[ExpressionLength];    // Temporary space for substring
    char* pstr=0;                     // Pointer to new string for return
    int numL = 0;                     // Count of left parentheses found
    int bufindex = index;              // Save starting value for index

    do {
        buffer[index-bufindex] = *(str+index);
        switch(buffer[index-bufindex]) {
            case ')': if(numL==0) {
                buffer[index-bufindex] = '\0'; // Replace ')' with '\0'
                ++index;
                pstr = new char[index-bufindex];

                if(!pstr) {
                    cout << "Memory allocation failed,"
                        << " program terminated.";
                    exit(1);
                }
                strcpy(pstr,buffer); // Copy substring to new memory
                return pstr;
            }
            else
                numL--; // Reduce count of '(' to be matched
            break;

            case '(':
                numL++; // Increase count of '(' to be matched
                break;
        }
    } while (*(str+index++) != '\0'); // Loop - don't overrun end of string

    cout << "Ran off the end of the expression, must be bad input." << endl;
    exit(1);

    return pstr;
}

```

```

double Parse (string Expression, const double AlloyFraction, const double Temperature)
{
    //Input area for expression to be evaluated
    char buffer[ExpressionLength] = {0};

    //Generally x(or X) is used for Alloy fraction
    //and T (or t) for Temperature
    string OCFileTokens("xXtT");
    string sFraction("0."), sTemperature("0.");

```

```

stringstream sstr;

sstr << Temperature;
getline(sstr,sTemperature);
sstr.clear();

sstr << AlloyFraction;
getline(sstr,sFraction);
sstr.clear();

if(Expression.empty()) {
    cout<<"\n\tBlank line found for equation";
    exit(1);
}

size_t LeftPos=0;

while( Expression.find_first_of(OCFileTokens, LeftPos) !=string::npos ) {
    LeftPos = Expression.find_first_of(OCFileTokens, LeftPos);
    if (toupper(Expression[LeftPos]) == 'X')
        Expression.replace (LeftPos++,1,sFraction);
    if (toupper(Expression[LeftPos]) == 'T')
        Expression.replace (LeftPos++,1,sTemperature);
}

for(int i=0; i<Expression.size();i++)
    buffer[i]=Expression[i];
buffer[i] = '\0';
Trim(buffer);

return Evaluate(buffer);
}

void OpticalConstants::SetNonDispersiveConstantsForStack(DielectricStack& DStack)
{
    vector <DielectricLayer> TempStack;
    bool bKey;
    int i=0, j;
    int key;
    string Path;

    if (DStack.StackOfLayers.size() == 0) {
        cout<<"\n\tEmpty Stack Found\n\t";
        exit(1);
    }
}

```

```

while(i<DStack.StackOfLayers.size()) {

    bKey = true;

    //Check: Constants already loaded?
    if (i != 0) {
        for (j=0; j<TempStack.size(); j++) {
            if (DStack.StackOfLayers[i] == TempStack[j]) {
                DStack.StackOfLayers[i] = TempStack[j];
                bKey = false;
                break;
            }
        }
    }

    if (bKey == true) {
        cout <<"\nLoad Optical Constants for Layer ["<<i+1<<"]:";
        cout <<"\n"<<DStack.StackOfLayers.at(i)<<endl;
        cout <<"\t1-> Optical Constants from Library"<<endl;
        cout <<"\t2-> User Input"<<endl;
        cout <<"\t3-> Layer Parameter File"<<endl;

        do
            key = atoi(AskUser("\tOption: ", false, true).c_str());
        while (key<1 || key>3);

        SetNonDispersiveConstantsForLayer(DStack.StackOfLayers.at(i), key, i);
        TempStack.push_back(DStack.StackOfLayers.at(i));
    }

    //Advance to next Layer in the Stack
    i++;
}
}

```

```

void OpticalConstants::SetNonDispersiveConstantsForLayer
    (class DielectricLayer& DLayer, int InputOption, int LayerIndex)
{
    string Path, FileName, str; stringstream sstr;

    switch (InputOption) {

        case 1: //Loading Optical Constants From User Defined or Default Library

            OutText("\tDefault Library: " + GetDefaultPath(), true);
            OutText("\tChange Path or hit <Enter> to continue: ");
            getline(cin, Path);

            if (Path.size() == 0)
                Path = GetDefaultPath();
            FileName = Path + DLayer.GetBaseMaterial() + " " +
                DLayer.GetBulkOrFilm() + ".dat";
            break;

        case 2: //Prompting user to input optical parameters
            OpticalConstantsFromUser(DLayer);
            return;

        case 3: //Loading Optical Constants From User defined files
            /*Expects the Layer Optical Parameter source in the
            same desination as *.exe*/
            //Insert Path within " " to modify path

            Path = " ";

            //creating default filename
            sstr<<LayerIndex+1;
            getline(sstr,str);
            FileName = "Layer " + str + ".dat";
            sstr.clear();

            //Prompting user to change default names
            OutText("\tDefault File: " + FileName, true);
            OutText("\tChange Name or hit <Enter> to continue: ", false);
            getline(cin,str);
            if (str.size() != 0)
                FileName = str;
            break;

        default: //Exit on Incorrect Option
            cout<<"\nInvalid Option to read Non Dispersive
            Optical Parameters"<<endl;
            exit(INVALID_OPTION);

    }
}

```

```

//If reading from file Proceed as follows
ifstream Infile;
size_t LeftPos;

//Initializing Phonon mode for reading
int MaxModes = DielectricLayer::PhononSize;
int ModeLO = -1;
int ModeTO = -1;

OpenFile(Infile, FileName);
Infile.clear();

double x(DLayer.AlloyFraction);
double T(DLayer.Temperature);

while(getline(Infile,str)) {
    if (str[0]!='%') {
        LeftPos = 0;

        //Dielectric Constants
        if (str.find("eps_h/eps_0", LeftPos) != string::npos) {
            LeftPos = str.find("eps_h/eps_0", LeftPos) +
                string("eps_h/eps_0 ").size();
            DLayer.HighFrequencyDielectricConstant
                = Parse(str.substr(LeftPos), x, T);
        }
        else if (str.find("eps_s/eps_0", LeftPos) != string::npos) {
            LeftPos = str.find("eps_s/eps_0", LeftPos) +
                string("eps_s/eps_0 ").size();
            DLayer.StaticDielectricConstant
                = Parse(str.substr(LeftPos), x, T);
        }

        //Effective Carrier Masses
        else if (DLayer.Dopant.GetType() == 'n' &&
            str.find("m_e/m_0", LeftPos) != string::npos) {
            LeftPos = str.find("m_e/m_0", LeftPos) + string("m_e/m_0 ").size();
            DLayer.Dopant.SetEffectiveMass(Parse(str.substr(LeftPos), x, T));
        }
        else if (DLayer.Dopant.GetType() == 'p' &&
            str.find("m_h/m_0", LeftPos) != string::npos) {
            LeftPos = str.find("m_h/m_0", LeftPos) + string("m_h/m_0 ").size();
            DLayer.Dopant.SetEffectiveMass(Parse(str.substr(LeftPos), x, T));
        }
    }
}

```

```

//Plasmon Parameters
else if (str.find("W_plasmon/cm^-1", LeftPos) != string::npos) {
    LeftPos = str.find("W_plasmon/cm^-1", LeftPos)
        + string("(W_plasmon/cm^-1)").size();
    DLayer.Plasmon.SetResonanceFrequency
        (Parse(str.substr(LeftPos), x, T));
}
else if (str.find("D_plasmon", LeftPos) != string::npos) {
    LeftPos = str.find("D_plasmon", LeftPos) +
        string("(D_plasmon)").size();
    DLayer.Plasmon.SetDamping(Parse(str.substr(LeftPos), x, T));
}

//TO-Phonon Parameters
else if (str.find("W_TO/cm^-1", LeftPos) != string::npos &&
    ModeTO <= MaxModes) {
    LeftPos = str.find("W_TO/cm^-1", LeftPos) +
        string("(W_TO/cm^-1)").size();
    DLayer.TO_Phonon[++ModeTO].
        SetResonanceFrequency(Parse(str.substr(LeftPos), x, T));
}
else if (str.find("S_TO", LeftPos) != string::npos &&
    ModeTO <= MaxModes){
    LeftPos = str.find("S_TO", LeftPos) + string("(S_TO)").size();
    DLayer.TO_Phonon.at(ModeTO).
        SetStrength(Parse(str.substr(LeftPos), x, T));
}
else if (str.find("D_TO/cm^-1", LeftPos) != string::npos &&
    ModeTO <= MaxModes) {
    LeftPos = str.find("D_TO/cm^-1", LeftPos) +
        string("(D_TO/cm^-1)").size();

    DLayer.TO_Phonon.at(ModeTO).
        SetDamping(Parse(str.substr(LeftPos), x, T));
}

//LO-Phonon Constants
else if (str.find("W_LO/cm^-1", LeftPos) != string::npos &&
    ModeLO <= MaxModes ) {
    LeftPos = str.find("W_LO/cm^-1", LeftPos) +
        string("(W_LO/cm^-1)").size();
    DLayer.LO_Phonon[++ModeLO].
        SetResonanceFrequency(Parse(str.substr(LeftPos), x, T));
}
else if (str.find("S_LO", LeftPos) != string::npos) {
    LeftPos = str.find("S_LO", LeftPos) + string("(S_LO)").size();
    DLayer.LO_Phonon.at(ModeLO).
        SetStrength(Parse(str.substr(LeftPos), x, T));
}

```

```

        else if (str.find("D_LO/cm^-1", LeftPos) != string::npos) {
            LeftPos = str.find("D_LO/cm^-1", LeftPos) +
                string("D_LO/cm^-1 ").size();

            DLayer.LO_Phonon.at(ModeLO).
                SetDamping(Parse(str.substr(LeftPos), x, T));
        }
    }
}
Infile.close();

//Remove comment on "ShowOpticalConstant" to Display Optical Constants
ShowOpticalConstants(cout, DLayer);
DLayer.SetPermittivityModel();
}

void OpticalConstants::OpticalConstantsFromUser(DielectricLayer& DLayer)
{
    int MaxModes = DielectricLayer::PhononSize;

    //Dielectric Constants
    cout<<"\n\tEnter High frequency Dielectric Constant: ";
    cin>>DLayer.HighFrequencyDielectricConstant;
    cout<<"\n\tEnter Static Dielectric Constant: ";
    cin>>DLayer.StaticDielectricConstant;

    double m; //Effective Carrier Mass
    cout<<"\n\tEnter Effective Mass (m/mo): ";
    cin>>m;
    DLayer.Dopant.SetEffectiveMass(m);

    cout<<"\n\tEnter Plasmon frequency (cm^-1): "; //Plasmon Constants
    cin>>DLayer.Plasmon.ResonanceFrequency;
    if (DLayer.Plasmon.ResonanceFrequency != 0) {
        cout<<"\n\tEnter Plasmon Broadening (cm^-1): ";
        cin>>DLayer.Plasmon.Damping;
    }

    int Mode = -1; //TO-Phonon Constants
    while (Mode<0 || Mode>MaxModes) {
        cout<<"\n\tEnter the number of modes for TO: ";
        cin>>Mode;

        if (Mode>MaxModes) {
            cout<<"\tNumber of modes exceeded limit "<<MaxModes;
            continue;
        }
    }
}

```



```

        for (int i=0; i<Mode; i++) {
            cout<<"\n\tTO Phonon ["<<i+1<<"]:";
            cin>>DLayer.TO_Phonon.at(i);
        }
    }

    //LO-Phonon Constants
    Mode = -1;
    while (Mode<0 || Mode>MaxModes) {
        cout<<"\n\tEnter the number of modes for LO: ";
        cin>>Mode;
        if (Mode>MaxModes) {
            cout<<"\tNumber of modes exceeded limit "<<MaxModes;
            continue;
        }

        for (int i=0; i<Mode; i++) {
            cout<<"\n\tLO Phonon ["<<i+1<<"]:";
            cin>>DLayer.LO_Phonon.at(i);
        }
    }
}

void OpticalConstants::ShowOpticalConstants(ostream& out, const DielectricLayer& DLayer)
{
    out    <<"Base material:"<<"\t"
           <<DLayer.BaseMaterial<< "("<<DLayer.BulkOrFilm<< ")" <<endl

           <<"Dopant:"<<"\t"<<DLayer.Dopant<<endl
           <<"Epsilon_High:"<<"\t"
           <<DLayer.HighFrequencyDielectricConstant <<endl
           <<"Epsilon_Static:"<<"\t"
           <<DLayer.StaticDielectricConstant <<endl
           <<"Effective Mass:"<<"\t"
           <<DLayer.Dopant.GetEffectiveMass()<<endl;

    if (DLayer.Plasmon.ResonanceFrequency != 0)
        out<<"\n\tPlasmon: " <<DLayer.Plasmon;

    for (int i=0; i<DielectricLayer::PhononSize; ++i) {
        if (DLayer.TO_Phonon[i].ResonanceFrequency != 0.)
            out<<"\n\tTO-Phonon["<<i+1<<"]:" <<DLayer.TO_Phonon[i];
        if (DLayer.LO_Phonon[i].ResonanceFrequency != 0.)
            out<<"\n\tLO-Phonon["<<i+1<<"]:" <<DLayer.LO_Phonon[i];
    }

    out<<endl;
}

```

```

//Class Ambient-Definition
#ifndef AMBIENT_H
#define AMBIENT_H

#include "DielectricLayer.h"
#include "FresnelCoefficientMatrix.h"

using namespace std;

class DielectricLayer;
class FresnelCoefficientMatrix;

COMPLEX POW(COMPLEX Base, COMPLEX Power);

class Ambient{
private:
    COMPLEX Permittivity;
    COMPLEX Permeability;
    COMPLEX RefractiveIndex;

    void SetRefractiveIndex (const COMPLEX& _Permittivity,
                             const COMPLEX& _Permeability);
    void SetRefractiveIndex (const Ambient& Medium);
public:
    Ambient::Ambient (void);
    Ambient::Ambient (string Message) {/* Used as Dummy. Message? Neglect*/}
    Ambient::Ambient (const COMPLEX& _Permittivity,
                      const COMPLEX& _Permeability)
        :Permittivity(_Permittivity), Permeability(_Permeability)
        {
            SetRefractiveIndex();
        }

    void SetPermeability      (const COMPLEX& _Permeability);
    void SetPermittivity      (const COMPLEX& _Permittivity);
    void SetRefractiveIndex   (void);
    COMPLEX GetPermittivity   (void){return Permittivity;};
    COMPLEX GetPermeability   (void){return Permeability;};
    COMPLEX GetRefractiveIndex (void){return RefractiveIndex;};

    FresnelCoefficientMatrix
    FresnelCoefficient (DielectricLayer& DLayer, const double& Wavenumber, const
                       double& IncidentAngle, const char& Polarization);
    void PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient,
                                Gnuplot& PFC, const double& IncidentAngle= 0.);
    void PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient,
                                double Wavenumber, Gnuplot& PFC);
};

#endif

```

```

//Implementation of class Ambient
#include "stdafx.h"
#include "Ambient.h"
#include "Constants.h"
#include "Gnuplot_i.h"

// temporary measure to fix the negative-base bug in MSVC Version 6.0 Enterprise edition.
COMPLEX POW(COMPLEX Base, COMPLEX Power)
{
    if ( (Base.imag()==0) && (Base.real() < 0) )
        return -I*pow(-Base.real(), Power);
    else
        return pow(Base, Power);
}

Ambient::Ambient(void)
{
    cout<<"\n\tWarning! Ambient Not Initialized"<<endl;
    cout<<"\tEnter the Relative Permittivity:\t";
    cin>>Permittivity;
    cout<<"\tEnter the Relative Permeability:\t";
    cin>>Permeability;
    SetRefractiveIndex();
}

void Ambient::SetRefractiveIndex (const COMPLEX& _Permittivity, const COMPLEX& _Permeability)
{
    RefractiveIndex = pow(_Permittivity*_Permeability, 0.5);
}

void Ambient::SetRefractiveIndex (const Ambient& Medium)
{
    SetRefractiveIndex(Medium.Permittivity, Medium.Permeability);
}

void Ambient::SetRefractiveIndex (void)
{
    SetRefractiveIndex(*this);
}

void Ambient::SetPermeability(const COMPLEX& _Permeability)
{
    Permeability = _Permeability;
    SetRefractiveIndex();
}

```

```

void Ambient::SetPermittivity(const COMPLEX& _Permittivity)
{
    Permittivity = _Permittivity;
    SetRefractiveIndex();
}

FresnelCoefficientMatrix
Ambient::FresnelCoefficient (DielectricLayer& DLayer, const double& _Wavenumber,
                           const double& Theta, const char& Polarization)
{
    FresnelCoefficientMatrix FCM;

    COMPLEX n_In, mu_In, e_In;
    COMPLEX n_Out, mu_Out, e_Out;

    //Retrieving Ambient Parameters
    e_In = GetPermittivity();
    mu_In = GetPermeability();
    n_In = POW(e_In*mu_In - e_In*mu_In*pow(sin(Theta),2), 0.5);

    //Retrieving Layer Parameters
    e_Out = DLayer.GetPermittivity(_Wavenumber);
    mu_Out = DLayer.GetPermeability(_Wavenumber);
    n_Out = POW(e_Out*mu_Out - e_In*mu_In*pow(sin(Theta),2), 0.5);

    //Amplitudes and Intensity for s-Polarization
    if(Polarization == 's') {
        //Electrical Components for TE-Mode
        FCM.rs.Electrical= (n_In/mu_In - n_Out/mu_Out)/(n_In/mu_In + n_Out/mu_Out);
        FCM.ts.Electrical= (2.*n_In/mu_In)/(n_In/mu_In + n_Out/mu_Out);

        //Magnetic Components for TE-Mode
        FCM.rs.Magnetic = -FCM.rs.Electrical;
        FCM.ts.Magnetic = (pow(e_Out*mu_In/(e_In*mu_Out), 0.5))* FCM.ts.Electrical;

        //Optical Field for TE-Mode
        FCM.Rs.Magnitude = norm(FCM.rs.Electrical);
        FCM.Ts.Magnitude = norm(FCM.ts.Electrical) *
                           real( (n_Out*mu_In)/(n_In*mu_Out) );
    }

    //Amplitudes and Intensity for p-Polarization
    else if (Polarization == 'p') {
        //Magnetic Components for for TM-Mode
        FCM.rp.Magnetic = (n_In/e_In - n_Out/e_Out)/(n_In/e_In + n_Out/e_Out);
        FCM.tp.Magnetic = (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

        //Electrical Components for for TM-Mode
        FCM.rp.Electrical = -FCM.rp.Magnetic;
    }
}

```

```

FCM.tp.Electrical = pow (mu_Out*e_In/(mu_In*e_Out), 0.5) *
                    (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

//Optical Field for TE-Mode
FCM.Rp.Magnitude = norm(FCM.rp.Magnetic);
FCM.Tp.Magnitude = norm(FCM.tp.Magnetic)* real( (n_Out*e_In)/(n_In*e_Out) );
}

else {
    cerr<<"\n\tUndefined Polarization"<<endl;
    exit(UNDEFINED_POLARIZATION);
}
return FCM;
}

```

```

void Ambient::
PlotFresnelCoefficient(DielectricLayer& DLayer, string FCoefficient, Gnuplot& PFC,
                      const double& Theta)
{
    double Wavenumber;
    FresnelCoefficientMatrix FCM;
    char Polarization;

    //Retrieving Polarization
    if ( FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 ||
        FCoefficient.compare("Rs") == 0 || FCoefficient.compare("Ts") == 0 ) {
        Polarization = 's';
    }

    else if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
        FCoefficient.compare("Rp") == 0 || FCoefficient.compare("Tp") == 0 ) {
        Polarization = 'p';
    }

    else {
        cout<<"\n\tUndefined Fresnel Coefficient"<<endl;
        exit(INCORRECT_POLARIZATION);
    }

    try {

        PFC.Out<<"Fresnel Coefficient:"<<FCoefficient<<endl
            <<"Incident/Emerging"<<endl
            <<"Refractive Index of Ambient: "<<this->RefractiveIndex <<endl
            <<DLayer <<endl <<endl
            <<"Incident Light = "
            <<Theta*180./pi<<" Deg ("<<Polarization<<" Polarized)"
            <<endl <<endl;
    }
}

```

```

if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
    FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 )

    PFC.Out<<"Wavenumber(cm^-1)\tElectrical-Amplitude\tElectrical-Phase
    Difference\tMagnetic-Amplitude\tMagnetic-Phase Difference"<<endl;
else
    PFC.Out<<"Wavenumber(cm^-1)\tOptical Field"<<endl;

vector<double> _Wavenumber;
vector<Wave> _Wave;
vector<Intensity> Int;
Wavenumber = PFC.LimitLower;

while (Wavenumber <= PFC.LimitUpper) {
    FCM = FresnelCoefficient(DLayer, Wavenumber, Theta, Polarization);
    _Wavenumber.push_back(Wavenumber);

    if (FCoefficient.compare("rs") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.rs<<endl;
        _Wave.push_back(FCM.rs);
    }
    else if (FCoefficient.compare("rp") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.rp<<endl;
        _Wave.push_back(FCM.rp);
    }
    else if (FCoefficient.compare("ts") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.ts<<endl;
        _Wave.push_back(FCM.ts);
    }
    else if (FCoefficient.compare("tp") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.tp<<endl;
        _Wave.push_back(FCM.tp);
    }
    else if (FCoefficient.compare("Rs") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.Rs<<endl;
        Int.push_back(FCM.Rs);
    }
    else if (FCoefficient.compare("Rp") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.Rp<<endl;
        Int.push_back(FCM.Rp);
    }
    else if (FCoefficient.compare("Ts") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.Ts<<endl;
        Int.push_back(FCM.Ts);
    }
    else if (FCoefficient.compare("Tp") == 0) {
        PFC.Out<<Wavenumber<<"\t"<<FCM.Tp<<endl;
        Int.push_back(FCM.Tp);
    }
    Wavenumber += PFC.SizeStep;
}

```

```

    }
    PFC.Out.close();

    if (_Wave.size() != 0)
        PFC.PlotWave(_Wavenumber, _Wave, FCoefficient);
    else
        PFC.PlotIntensity(_Wavenumber, Int, FCoefficient);
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}

void Ambient::
PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient, double Wavenumber,
                        Gnuplot& PFC)
{
    //Angles in degrees
    double Theta;
    FresnelCoefficientMatrix FCM;
    char Polarization;

    //Retrieving Polarization
    if ( FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 ||
        FCoefficient.compare("Rs") == 0 || FCoefficient.compare("Ts") == 0 ) {
        Polarization = 's';
    }

    else if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
        FCoefficient.compare("Rp") == 0 || FCoefficient.compare("Tp") == 0 ) {
        Polarization = 'p';
    }
    else {
        cout<<"\n\tUndefined Fresnel Coefficient"<<endl;
        exit(UNDEFINED_FRESNEL_COEF);
    }

    try {
        PFC.Out<<"Fresnel Coefficient:"<<FCoefficient<<endl
            <<"Incident/Emerging"<<endl
            <<"Refractive Index of Ambient: "<<this->RefractiveIndex<<endl
            <<DLayer <<endl<<endl
            <<"Incident Light = "
            <<Wavenumber<<" cm^-1 ("<<Polarization<<" Polarized)" <<endl<<endl;
    }
}

```

```

if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
    FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 )

    PFC.Out<<"Incident Angle (Deg)\tReal(Electrical)\t Imag(Electrical)
        \tReal(Magnetic)\tImag(Magnetic)" <<endl;
else
    PFC.Out<<"Incident Angle (Deg)\tOptical Field"<<endl;

vector<double> _Theta;
vector<Wave> _Wave;
vector<Intensity> Int;
Theta = PFC.LimitLower;

while (Theta <= PFC.LimitUpper) {

    FCM = FresnelCoefficient(DLayer, Wavenumber, pi*Theta/180, Polarization);
    _Theta.push_back(Theta);

    if (FCoefficient.compare("rs") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.rs<<endl;
        _Wave.push_back(FCM.rs);
    }
    else if (FCoefficient.compare("rp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.rp<<endl;
        _Wave.push_back(FCM.rp);
    }
    else if (FCoefficient.compare("ts") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.ts<<endl;
        _Wave.push_back(FCM.ts);
    }
    else if (FCoefficient.compare("tp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.tp<<endl;
        _Wave.push_back(FCM.tp);
    }
    else if (FCoefficient.compare("Rs") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Rs<<endl;
        Int.push_back(FCM.Rs);
    }
    else if (FCoefficient.compare("Rp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Rp<<endl;
        Int.push_back(FCM.Rp);
    }
    else if (FCoefficient.compare("Ts") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Ts<<endl;
        Int.push_back(FCM.Ts);
    }
    else if (FCoefficient.compare("Tp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Tp<<endl;
        Int.push_back(FCM.Tp);
    }
}

```



```
        Theta += PFC.SizeStep;
    }

    PFC.Out.close();

    if (_Wave.size() != 0)
        PFC.PlotWave(_Theta, _Wave, FCoefficient);
    else
        PFC.PlotIntensity(_Theta, Int, FCoefficient);
}

catch (GnuplotException ge) {
    cout<<ge.what() <<endl;
}
}
```

```

//Declaration for the DielectricLayer Class
#ifndef LAYER_H
#define LAYER_H

#include <windows.h>
#include <conio.h>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include <math.h>
#include <complex>

#include "Gnuplot_i.h"
#include "OpticalConstants.h"
#include "Constants.h"
#include "Carrier.h"
#include "Oscillator.h"
#include "FresnelCoefficientMatrix.h"
#include "Date.h"
#include "Time.h"

using namespace std;

string Tolower (string& Word);

//Forward Declaration
class Carrier;
class FreeOscillator;
class BoundOscillator;
class OpticalConstants;
enum MaterialOrder;
class OpticalConstants;
class DielectricLayer;

class DielectricLayer
{
    static const int PhononSize;
    static const BoundOscillator BadOscillator;

    //Layer Properties
private:
    string    BulkOrFilm;
    string    BaseMaterial;
    double    AlloyFraction;
    double    Thickness;

```

```

/*Data Encapsulation of the "Carrier types" and "Oscillator types" are maintained through
private members in their respective classes. For safety, Replacing and Changing are not
allowed outside the DielectricLayer:: scope.*/

public:
    Carrier Dopant;
    Carrier Defect;
    FreeOscillator Plasmon;
    vector <BoundOscillator> TO_Phonon;
    vector <BoundOscillator> LO_Phonon;

private:
    double BandGap;
    double Temperature;
    string PlasmonModel;
    string PhononModel;
    string Label;

    //Non Dispersive Members
    double StaticDielectricConstant;
    double HighFrequencyDielectricConstant;
    double DebyeTemperature;

    //Dispersive Members
    double Wavenumber;
    double PropagatingAngle;
    COMPLEX Permittivity;
    COMPLEX Permeability;
    COMPLEX PermittivityForCarrierGeneration;
    COMPLEX RefractiveIndex;
    double AbsorptionCoefficient;
    double SkinDepth;

    void SetPropagatingAngle (const double& _Wavenumber,
                              const double& Refraction_Index);
    void SetPermittivity (const double& _Wavenumber);
    void SetPermeability (const double& _Wavenumber);
    void SetRefractiveIndex (const double& _Wavenumber);
    void SetAbsorptionCoefficient (const double& _Wavenumber);
    void SetSkinDepth (const double& _Wavenumber);

public:
    //Default Constructor
    DielectricLayer (string Bulk_Or_Film, double Alloy_Fraction)
        :BulkOrFilm(Bulk_Or_Film), AlloyFraction(Alloy_Fraction){};
    DielectricLayer (string Bulk_Or_Film="UnSpecified",
        string Base_Material="UnSpecified",
        double _Thickness=0., char Dopant_Type='?',
        double Dopant_Density=0.,
        string Layer_Label = "Unspecified", double Alloy_Fraction=0.);

```

```

~DielectricLayer(){};

int      Material(string _Material);//This will return the order as enumerated
void     Erase(void);
bool     DielectricLayer::operator==(DielectricLayer& DLayer) const;
bool     DielectricLayer::operator!=(DielectricLayer& DLayer) const;
void     SetBulkOrFilm (string Bulk_Or_Film)
        {BulkOrFilm = Tolower(Bulk_Or_Film)};

void     SetBaseMaterial (string Base_Material)
        {BaseMaterial = Tolower(Base_Material)};

void     SetAlloyFraction (const double Alloy_Fraction)
        {AlloyFraction = Alloy_Fraction};

void     SetThickness (const double _Thickness)
        {Thickness = _Thickness};

void     SetDopant (const Carrier& _Dopant)
        {Dopant = _Dopant};

void     SetDefect (const Carrier& _Defect)
        {Defect = _Defect};

void     SetLabel (string _Label)
        {Label = Tolower(_Label)};

void     SetPermittivityModel(void);

void     SetPhononModel (string Phonon_Model)
        {PhononModel = Tolower(Phonon_Model)};

void     SetBandGap (const double Band_Gap)
        {BandGap = Band_Gap};

void     SetTemperature (const double& _Temperature)
        {Temperature = _Temperature};

void     SetPlasmon (const FreeOscillator& Oscillator)
        {Plasmon = Oscillator};

void     SetTO_Phonon      (const int Index, const BoundOscillator& Oscillator);

void     SetLO_Phonon      (const int Index, const BoundOscillator& Oscillator);

void     SetOpticalConstants (const double& Wavenumber);

string   SetBandGap        (void);

```

```

void      SetRefractiveIndex      (const COMPLEX& _Permittivity,
                                   const COMPLEX& _Permeability);

string     GetBulkOrFilm          (void) const    {return BulkOrFilm;};
string     GetBaseMaterial        (void) const    {return BaseMaterial;};
double     GetAlloyFraction        (void) const    {return AlloyFraction;};
double     GetThickness            (void) const    {return Thickness;};
Carrier     GetDopant              (void) const    {return Dopant;};
Carrier     GetDefect              (void) const    {return Defect;};
string     GetLabel                (void) const    {return Label;};
string     GetPlasmonModel         (void) const    {return PlasmonModel;};
string     GetPhononModel         (void) const    {return PhononModel;};
double     GetBandGap              (void) const    {return BandGap;};
double     GetTemperature          (void) const    {return Temperature;};

COMPLEX     GetPermittivityForCarrierGeneration  (void)
                                   {return PermittivityForCarrierGeneration;};

FreeOscillator  GetPlasmon      (void) const          {return Plasmon;};
BoundOscillator GetTO_Phonon    (const int Index) const;
BoundOscillator GetLO_Phonon    (const int Index) const;

double         GetWavenumber          (void) {return Wavenumber;};
double         GetPropagatingAngle     (void) {return PropagatingAngle;};
COMPLEX        GetPermittivity         (double _Wavenumber);
COMPLEX        GetPermeability         (double _Wavenumber);
COMPLEX        GetRefractiveIndex      (void) {return RefractiveIndex;};
double         GetAbsorptionCoefficient (void)
                                   {return AbsorptionCoefficient;};

double  GetSkinDepth (void) {return SkinDepth;};

FresnelCoefficientMatrix
FresnelCoefficient (DielectricLayer& DLayer, const double& _Wavenumber,
                   const double& IncidentAngle, const char& Polarization);

FresnelCoefficientMatrix
FresnelCoefficient (Ambient& AmbientMedium, const double& _Wavenumber,
                   const double& IncidentAngle, const char& Polarization);

FresnelCoefficientMatrix
FresnelCoefficient (DielectricLayer& DLayer, Ambient& IncidentMedium,
                   const double& _Wavenumber, const double& IncidentAngle,
                   const char& Polarization);

FresnelCoefficientMatrix
FresnelCoefficient (Ambient& EmergingMedium, Ambient& IncidentMedium,
                   const double& _Wavenumber, const double& IncidentAngle,
                   const char& Polarization);

```

```

void PlotPermittivity          (Gnuplot& GP);
void PlotPermeability          (Gnuplot& GP);
void PlotRefractiveIndex        (Gnuplot& GP);
void PlotAbsorptionCoefficient   (Gnuplot& GP);
void PlotSkinDepth              (Gnuplot& GP);

//FresnelCoefficients for DielectricLayer/DielectricLayer
void PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient,
                             Gnuplot& PFC, const double& IncidentAngle= 0.);

void PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient,
                             const double& Wavenumber, Gnuplot& PFC);

//FresnelCoefficients for DielectricLayer/Ambient

void PlotFresnelCoefficient (Ambient& AmbientMedium, string FCoefficient,
                             Gnuplot&PFC, const double& IncidentAngle= 0.);

void PlotFresnelCoefficient (Ambient& AmbientMedium, string FCoefficient,
                             const double& Wavenumber, Gnuplot& PFC);


friend ostream& operator<<(ostream& out, const DielectricLayer& DLayer);
friend istream& operator>>(istream& in, DielectricLayer& DLayer);

friend void OpticalConstants::
    SetNonDispersiveConstantsForLayer
    (DielectricLayer& DLayer, int LayerIndex, int InputOption);

friend void OpticalConstants::
    OpticalConstantsFromUser (DielectricLayer& DLayer);

friend void OpticalConstants::
    ShowOpticalConstants (ostream& out, const DielectricLayer& DLayer);
};

#endif

```

```

//Implimentation of Class DielectricLayer
#include "stdafx.h"
#include "DielectricLayer.h"
#include "Carrier.h"
#include "Ambient.h"

//Exit Indices
#define OnBadPhononIndex 1

//Non Member Functions
extern string Tolower (string& Word)
{
    transform(Word.begin(), Word.end(), Word.begin(), tolower);
    return Word;
}

enum MaterialOrder {
    //Binaries

    //Arsenides
    GaAs,
    InAs,
    //Nitrides
    GaN,
    InN,
    //Antimonides
    GaSb,
    InSb,
    //substrate materials not listed above
    Sapphire,
    /*additional binaries should be added before this line. Remember to modify the relevant portions
    in the header*/

    //Ternaries
    //Arsenides
    AlGaAs,
    InGaAs,
    //Nitrides
    AlGaN,
    GaMnN,
    //Antimonides
    InGaSb
};

```

```

//Maximumm Limit can be changed here
const int DielectricLayer::
PhononSize =10;

//BadOscillator Assignment can be changed here
static const BoundOscillator BadOscillator = BoundOscillator(-1, -1, -1);

int DielectricLayer::
Material(string _Material)
{
    if (_Material.compare("GaAs") == 0)
        return GaAs;
    else if (_Material.compare("InAs") == 0)
        return InAs;
    else if (_Material.compare("GaN") == 0)
        return GaN;
    else if (_Material.compare("InN") == 0)
        return InN;
    else if (_Material.compare("GaSb") == 0)
        return GaSb;
    else if (_Material.compare("InSb") == 0)
        return InSb;
    else if (_Material.compare("Sapphire") == 0)
        return Sapphire;
    else if (_Material.compare("AlGaAs") == 0)
        return AlGaAs;
    else if (_Material.compare("AlGaN") == 0)
        return AlGaN;
    else if (_Material.compare("GaMnN") == 0)
        return GaMnN;

    cerr<<"\n\t"<<BaseMaterial<<" is not in the material list";
    exit(0);
}

DielectricLayer::
DielectricLayer (string Bulk_Or_Film, string Base_Material, double _Thickness, char Dopant_Type,
                double Doping_Density, string _Label, double Alloy_Fraction)
:Dopant(tolower(Dopant_Type), Doping_Density),
Defect(),
Plasmon(),
TO_Phonon(PhononSize,BoundOscillator()),
LO_Phonon(PhononSize,BoundOscillator()),
Thickness(_Thickness),
AlloyFraction(Alloy_Fraction),
BulkOrFilm(Tolower(Bulk_Or_Film)),
BaseMaterial(Base_Material),
Label(Tolower(_Label))

```



```

{
    Temperature           = 0.;
    PlasmonModel           = "unknown";
    PhononModel            = "unknown";
    DebyeTemperature       = 0.;
    StaticDielectricConstant = 0.;
    HighFrequencyDielectricConstant = 0.;
    PropagatingAngle       = 0.;
    Permittivity           = 0.;
    Permeability           = 0.;
    RefractiveIndex        = 0.;
    AbsorptionCoefficient   = 0.;
    SkinDepth              = 0.;
}

```

```

std::ostream&
operator<<(ostream &out, const DielectricLayer& DLayer)
{
    return out
        <<DLayer.BulkOrFilm <<"\t"
        <<DLayer.BaseMaterial <<"\t"
        <<DLayer.Thickness <<" cm\t"
        <<DLayer.Dopant <<"\t"
        <<DLayer.Label <<"\t"
        <<DLayer.AlloyFraction;
}

```

```

std::istream&
operator>>(istream &in, DielectricLayer& DLayer)
{
    in
        >>DLayer.BulkOrFilm
        >>DLayer.BaseMaterial
        >>DLayer.Thickness
        >>DLayer.Dopant
        >>DLayer.Label
        >>DLayer.AlloyFraction;

    DLayer.BulkOrFilm = Tolower(DLayer.BulkOrFilm);
    DLayer.Label = Tolower(DLayer.Label);

    return in;
}

```

```

bool DielectricLayer::operator==(DielectricLayer& DLayer) const
{
    return ( DLayer.BulkOrFilm.compare(BulkOrFilm) == 0 &&
            DLayer.BaseMaterial.compare(BaseMaterial) == 0 &&
            DLayer.Thickness == Thickness &&
            DLayer.Dopant == Dopant &&
            DLayer.AlloyFraction == AlloyFraction);
}

bool DielectricLayer::operator!=(DielectricLayer& DLayer) const
{
    return !(operator==(DLayer));
}

void DielectricLayer::Erase(void)
{
    //Erase Private Members
    BulkOrFilm.erase();
    BaseMaterial.erase();
    Thickness = 0.;
    Label.erase();
    AlloyFraction = 0.;
    Dopant.Erase();
    Defect.Erase();
    Plasmon.Erase();

    //Erase Public Members
    StaticDielectricConstant = 0.;
    HighFrequencyDielectricConstant = 0.;
    Temperature = 0.;

    for(int i=0; i<DielectricLayer::PhononSize; i++) {
        TO_Phonon.at(i).Erase();
        LO_Phonon.at(i).Erase();
    }

    BandGap = 0.;
    DebyeTemperature = 0.;
}

void DielectricLayer::SetPermittivityModel(void)
{
    int Key;

    if (Plasmon.ResonanceFrequency != 0) {
        cout<<"\n\tContribution from Carrier Plasma:";
        cout<<"\n\t(1) Drude Model";
        cout<<"\n\t(2) Linhard Mermin Model";
    }
}

```

```

        do {
            cout<<"\n\tOption: ";
            cin>>Key;
        } while (Key != 1 && Key != 2);

        if (Key==1)
            PlasmonModel = "Drude";
        else
            PlasmonModel = "Linhard Mermin";
    }

    cout<<"\n\tContribution from Lattice Vibrations";
    cout<<"\n\t(1) Additive Multioscillator";
    cout<<"\n\t(2) Factorized Multioscillator";

    do {
        cout<<"\n\tOption: ";
        cin>>Key;
    } while (Key != 1 && Key != 2);

    if (Key==1)
        PhononModel = "Additive Multioscillator";
    else
        PhononModel = "Factorized Multioscillator";
}

void DielectricLayer::SetTO_Phonon (const int Index, const BoundOscillator& Oscillator)
{
    if (Index<0 || Index>PhononSize) {
        cerr<<"Bad Index encountered at SetTO_Phonon()"<<endl;
        cerr<<"Phonon Index should be in 0-"<<PhononSize<<endl;
        exit(OnBadPhononIndex);
    }
    else
        TO_Phonon.at(Index) = Oscillator;
}

void DielectricLayer::SetLO_Phonon (const int Index, const BoundOscillator& Oscillator)
{
    if (Index<0 || Index>PhononSize) {
        cerr<<"Bad Index encountered at SetLO_Phonon()"<<endl;
        cerr<<"Phonon Index should be in 0-"<<PhononSize<<endl;
        exit(OnBadPhononIndex);
    }
    else
        LO_Phonon.at(Index) = Oscillator;
}

```

```

BoundOscillator DielectricLayer::GetTO_Phonon (const int Index) const
{
    if (Index<0 || Index>PhononSize) {
        cerr<<"Bad Index encountered at GetTO_Phonon()"<<endl;
        cerr<<"Phonon Index should be in 0-"<<PhononSize<<endl;
        exit(OnBadPhononIndex);
    }

    return TO_Phonon.at(Index);
}

```

```

BoundOscillator DielectricLayer::GetLO_Phonon (const int Index) const
{
    if (Index<0 || Index>PhononSize) {
        cerr<<"Bad Index encountered at GetLO_Phonon()"<<endl;
        cerr<<"Phonon Index should be in 0-"<<PhononSize<<endl;
        exit(OnBadPhononIndex);
    }

    return LO_Phonon.at(Index);
}

```

```

void DielectricLayer::SetPermeability(const double& _Wavenumber)
{
    //For Non-Magnetic Material Permiability is not dispersive
    //For Magnetic Material Modify the following to the
    //relevant dispersion form

    if (Wavenumber == _Wavenumber)
        return;

    Permeability = COMPLEX(1.,0);
}

```

```

void DielectricLayer::SetPermittivity(const double& _Wavenumber)
{
    if (Wavenumber == _Wavenumber)
        return;

    //For Doped or Undoped Semiconductors
    //Intraband Transition region

    //Evaluating Phonon Contribution
    //-----
    COMPLEX PhononContribution(HighFrequencyDielectricConstant);

    //MultiOscillator Additive Model

```

```

if( PhononModel.compare("Additive Multioscillator") == 0 ) {
    for(int i=0; i<PhononSize; i++) {
        PhononContribution +=
            TO_Phonon.at(i).Strength*pow(TO_Phonon.at(i).ResonanceFrequency,2)/
            COMPLEX(pow(TO_Phonon.at(i).ResonanceFrequency,2) -
                _Wavenumber*_Wavenumber, -_Wavenumber*TO_Phonon.at(i).Damping);
    }
}

//MultiOscillator Factorial Model
else if (PhononModel.compare("Factorized Multioscillator") ==0 ) {
    for(int i=0; i<PhononSize; i++) {
        if ( (TO_Phonon.at(i).ResonanceFrequency != 0. &&
            LO_Phonon.at(i).ResonanceFrequency == 0.) ||

            (LO_Phonon.at(i).ResonanceFrequency != 0. &&
            TO_Phonon.at(i).ResonanceFrequency == 0.) ) {

            cout <<"\n\tIncorrect LO-TO combination found for"
                <<BulkOrFilm<<" "<<BaseMaterial<<endl;
            exit(INCORRECT_LO_TO_PAIRS);
        }

        PhononContribution *=
            COMPLEX(pow(LO_Phonon.at(i).ResonanceFrequency,2)
                - _Wavenumber*_Wavenumber, -
                _Wavenumber*LO_Phonon.at(i).Damping)/

            COMPLEX(pow(TO_Phonon.at(i).ResonanceFrequency,2) -
                _Wavenumber*_Wavenumber, -
                _Wavenumber*TO_Phonon.at(i).Damping);
    }
}

//Exiting on Undefined Phonon Model
else if (PhononModel.compare("unknown") ==0 ) {
    cerr<<"\nPhonon Model is not Defined"<<endl;
    exit(15);
};

//Evaluating Plasmon Contribution
//-----
COMPLEX PlasmonContribution(HighFrequencyDielectricConstant);

//Drude-Lorentz Model
if( PlasmonModel.compare("Drude") ==0 )
    PlasmonContribution *= -pow(Plasmon.ResonanceFrequency, 2)/
        COMPLEX(_Wavenumber*_Wavenumber, _Wavenumber*Plasmon.Damping);

//Lindhard-Mermin Model

```

```

else if( PlasmonModel.compare("Linhard Mermin") ==0 ){
    cout<<"\n\tLinhard Model is not coded yet"<<endl;
    exit(EXIT);
}
else
    PlasmonContribution = 0;

//Total contribution
//Contribution from the deepvalence electrons has been added to the phonon model for simplicity
Permittivity = PhononContribution + PlasmonContribution;
PermittivityForCarrierGeneration = HighFrequencyDielectricConstant + PlasmonContribution;
}

void DielectricLayer::SetRefractiveIndex(const double& _Wavenumber)
{
    if (Wavenumber == _Wavenumber)
        return;

    SetPermittivity(_Wavenumber);
    SetPermeability(_Wavenumber);
    RefractiveIndex = pow(Permittivity*Permeability,0.5);
}

void DielectricLayer::SetAbsorptionCoefficient(const double& _Wavenumber)
{
    if (Wavenumber == _Wavenumber)
        return;

    SetRefractiveIndex(_Wavenumber);
    AbsorptionCoefficient = 4*pi*_Wavenumber*imag(RefractiveIndex);
}

void DielectricLayer::SetSkinDepth (const double& _Wavenumber)
{
    if (Wavenumber == _Wavenumber)
        return;

    SetAbsorptionCoefficient(_Wavenumber);
    SkinDepth = 2./AbsorptionCoefficient;
}

```

```

void DielectricLayer::SetPropagatingAngle
    (const double& _Wavenumber, const double& Refractive_Index)
{
    SetRefractiveIndex(_Wavenumber);
    PropagatingAngle = asin(Refractive_Index/RefractiveIndex.real());
}

void DielectricLayer::SetOpticalConstants (const double& _Wavenumber)
{
    //The Order of Function calls inside shouldnt be changed

    if (Wavenumber == _Wavenumber)
        return;

    SetPermittivity(_Wavenumber);
    SetPermeability(_Wavenumber);
    SetRefractiveIndex(_Wavenumber);
    SetAbsorptionCoefficient(_Wavenumber);
    SetSkinDepth(_Wavenumber);

    Wavenumber = _Wavenumber;
}

COMPLEX DielectricLayer::GetPermittivity (double _Wavenumber)
{
    SetOpticalConstants(_Wavenumber);
    return Permittivity;
}

COMPLEX DielectricLayer::GetPermeability (double _Wavenumber)
{
    SetOpticalConstants(_Wavenumber);
    return Permeability;
}

void DielectricLayer::PlotPermeability (Gnuplot& GP)
{
    double _Wavenumber = GP.LimitLower;

    while (_Wavenumber <= GP.LimitUpper) {
        SetPermeability(_Wavenumber);
        GP.Out<<_Wavenumber<<"\t"<<real(Permeability)<<"\t"<<imag(Permeability)<<endl;
        _Wavenumber += GP.SizeStep;
    }
}

```

```

void DielectricLayer::PlotPermittivity(Gnuplot& GP)
{
    vector<double> _Wavenumber;
    vector<COMPLEX> _Permittivity;
    double Wavenumber = GP.LimitLower;

    try {
        GP.Out << *this << endl
            << endl << endl;

        GP.Out << "Wavenumber (cm^-1)\tReal(Permittivity)\tImag(Permittivity)" << endl;
        while (Wavenumber <= GP.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            SetPermittivity(Wavenumber);
            _Permittivity.push_back(Permittivity);
            GP.Out << Wavenumber << "\t" << real(Permittivity) << "\t"
                << imag(Permittivity) << endl;
            Wavenumber += GP.SizeStep;
        }

        GP.Out.close();
        GP.PlotComplex(_Wavenumber, _Permittivity, "Epsilon");
    }

    catch (GnuplotException ge) {
        cout << ge.what() << endl;
    }
}

void DielectricLayer::PlotRefractiveIndex(Gnuplot& GP)
{
    vector<double> _Wavenumber;
    vector<COMPLEX> _RefractiveIndex;
    double Wavenumber = GP.LimitLower;

    try {
        GP.Out << *this << endl
            << endl << endl;

        GP.Out << "Wavenumber (cm^-1)\tRefractive Index\tExtinction Coefficient" << endl;
        while (Wavenumber <= GP.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            SetRefractiveIndex(Wavenumber);
            _RefractiveIndex.push_back(RefractiveIndex);
            GP.Out << Wavenumber << "\t" << real(RefractiveIndex) << "\t"
                << imag(RefractiveIndex) << endl;
            Wavenumber += GP.SizeStep;
        }
    }
}

```



```

        GP.Out.close();
        GP.PlotComplex(_Wavenumber,_RefractiveIndex,"RefIndex");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

void DielectricLayer::PlotAbsorptionCoefficient(Gnuplot& GP)
{
    vector <double> _Wavenumber;
    vector <double> Absorption_Coefficient;
    double Wavenumber = GP.LimitLower;

    try {
        GP.Out << *this<<endl
            <<endl<<endl;

        GP.Out<<"Wavenumber (cm^-1)\tAbsorption Coefficient (cm^-1)"<<endl;
        while (Wavenumber <= GP.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            SetAbsorptionCoefficient(Wavenumber);
            Absorption_Coefficient.push_back(AbsorptionCoefficient);
            GP.Out<<Wavenumber<<"\t"<<AbsorptionCoefficient<<endl;
            Wavenumber += GP.SizeStep;
        }

        GP.Out.close();
        GP.plot_xy(_Wavenumber,Absorption_Coefficient,"RefIndex");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

void DielectricLayer::PlotSkinDepth(Gnuplot& GP)
{
    vector <double> _Wavenumber;
    vector <double> Skin_Depth;
    double Wavenumber = GP.LimitLower;

    try {
        GP.Out << *this<<endl
            <<endl<<endl;

        GP.Out<<"Wavenumber (cm^-1)\tSkin Depth (cm)"<<endl;

```

```

        while (Wavenumber <= GP.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            SetSkinDepth(Wavenumber);
            Skin_Depth.push_back(SkinDepth);
            GP.Out<<Wavenumber<<"\t"<<SkinDepth<<endl;
            Wavenumber += GP.SizeStep;
        }

        GP.Out.close();
        GP.plot_xy(_Wavenumber,Skin_Depth,"RefIndex");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

//Used to Calculate the FCM for Two DielectricLayer Interface
FresnelCoefficientMatrix DielectricLayer::
FresnelCoefficient (DielectricLayer& DLayer, const double& _Wavenumber,
                    const double& Theta, const char& Polarization)
{
    FresnelCoefficientMatrix FCM;
    COMPLEX n_In, mu_In, e_In;
    COMPLEX n_Out, mu_Out, e_Out;

    //Retrieving In Layer Parameters
    e_In = GetPermittivity(_Wavenumber);
    mu_In = GetPermeability(_Wavenumber);
    n_In = POW(e_In*mu_In - e_In*mu_In*pow(sin(Theta),2), 0.5);

    //Retrieving Out Layer Parameters
    e_Out = DLayer.GetPermittivity(_Wavenumber);
    mu_Out = DLayer.GetPermeability(_Wavenumber);
    n_Out = POW(e_Out*mu_Out - e_In*mu_In*pow(sin(Theta),2), 0.5);

    //Amplitudes and Intensity for s-Polarization
    if(Polarization == 's') {
        //Electrical Components for TE-Mode
        FCM.rs.Electrical = (n_In/mu_In - n_Out/mu_Out)/(n_In/mu_In + n_Out/mu_Out);
        FCM.ts.Electrical = (2.*n_In/mu_In)/(n_In/mu_In + n_Out/mu_Out);

        //Magnetic Components for TE-Mode
        FCM.rs.Magnetic = -FCM.rs.Electrical;
        FCM.ts.Magnetic = (pow(e_Out*mu_In/(e_In*mu_Out), 0.5))* FCM.ts.Electrical;
    }
}

```

```

        //Optical Field for TE-Mode
        FCM.Rs.Magnitude      = norm(FCM.rs.Electrical);
        FCM.Ts.Magnitude      = norm(FCM.ts.Electrical) *
                                real( (n_Out*mu_In)/(n_In*mu_Out) );
    }

    //Amplitudes and Intensity for p-Polarization
    else if (Polarization == 'p') {
        //Magnetic Components for for TM-Mode
        FCM.rp.Magnetic = (n_In/e_In - n_Out/e_Out)/(n_In/e_In + n_Out/e_Out);
        FCM.tp.Magnetic = (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

        //Electrical Components for for TM-Mode
        FCM.rp.Electrical = -FCM.rp.Magnetic;
        FCM.tp.Electrical = pow (mu_Out*e_In/(mu_In*e_Out), 0.5)*
                                (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

        //Optical Field for TE-Mode
        FCM.Rp.Magnitude = norm(FCM.rp.Magnetic);
        FCM.Tp.Magnitude = norm(FCM.tp.Magnetic)* real( (n_Out*e_In)/(n_In*e_Out) );
    }

    else {
        cerr<<"\n\tUndefined Polarization"<<endl;
        exit(UNDEFINED_POLARIZATION);
    }

    return FCM;
}

FresnelCoefficientMatrix DielectricLayer::
FresnelCoefficient (Ambient& AmbientMedium, const double& _Wavenumber, const double& Theta,
                    const char& Polarization)
{
    FresnelCoefficientMatrix FCM;
    COMPLEX n_In, mu_In, e_In;
    COMPLEX n_Out, mu_Out, e_Out;

    e_In = GetPermittivity(_Wavenumber);
    mu_In = GetPermeability(_Wavenumber);
    n_In = POW(e_In*mu_In - e_In*mu_In*pow(sin(Theta),2), 0.5);

    e_Out = AmbientMedium.GetPermittivity();
    mu_Out = AmbientMedium.GetPermeability();
    n_Out = POW(e_Out*mu_Out - e_In*mu_In*pow(sin(Theta),2), 0.5);

```

```

//Amplitudes and Intensity for s-Polarization
if(Polarization == 's') {
    //Electrical Components for TE-Mode
    FCM.rs.Electrical = (n_In/mu_In - n_Out/mu_Out)/(n_In/mu_In + n_Out/mu_Out);
    FCM.ts.Electrical = (2.*n_In/mu_In)/(n_In/mu_In + n_Out/mu_Out);

    //Magnetic Components for TE-Mode
    FCM.rs.Magnetic = -FCM.rs.Electrical;
    FCM.ts.Magnetic = (pow(e_Out*mu_In/(e_In*mu_Out), 0.5))* FCM.ts.Electrical;

    //Optical Field for TE-Mode
    FCM.Rs.Magnitude = norm(FCM.rs.Electrical);
    FCM.Ts.Magnitude = norm(FCM.ts.Electrical) *
        real( (n_Out*mu_In)/(n_In*mu_Out) );
}

//Amplitudes and Intensity for p-Polarization
else if (Polarization == 'p') {
    //Magnetic Components for for TM-Mode
    FCM.rp.Magnetic = (n_In/e_In - n_Out/e_Out)/(n_In/e_In + n_Out/e_Out);
    FCM.tp.Magnetic = (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

    //Electrical Components for for TM-Mode
    FCM.rp.Electrical = -FCM.rp.Magnetic;
    FCM.tp.Electrical = pow (mu_Out*e_In/(mu_In*e_Out), 0.5)*
        (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

    //Optical Field for TE-Mode
    FCM.Rp.Magnitude = norm(FCM.rp.Magnetic);
    FCM.Tp.Magnitude = norm(FCM.tp.Magnetic)* real( (n_Out*e_In)/(n_In*e_Out) );
}

else {
    cerr<<"\n\tUndefined Polarization"<<endl;
    exit(UNDEFINED_POLARIZATION);
}

return FCM;
}

void DielectricLayer::
PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient, Gnuplot& PFC,
    const double& Theta)
{
    double Wavenumber;
    FresnelCoefficientMatrix FCM;
    char Polarization;

```

```

//Retrieving Polarization
if ( FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 ||
    FCoefficient.compare("Rs") == 0 || FCoefficient.compare("Ts") == 0 ) {
    Polarization = 's';
}

else if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
    FCoefficient.compare("Rp") == 0 || FCoefficient.compare("Tp") == 0 ) {
    Polarization = 'p';
}

else {
    cout<<"\n\tUndefined Fresnel Coefficient"<<endl;
    exit(UNDEFINED_POLARIZATION);
}

try {

    PFC.Out<<"Fresnel Coefficient:" <<FCoefficient<<endl
        <<"Incident/Emerging" <<endl
        <<*this <<endl
        <<DLayer <<endl<<endl
        <<"Incident Light = "
        <<Theta*180./pi<<" Deg ("<<Polarization<<" Polarized)"
        <<endl<<endl;

    if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
        FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 )

        PFC.Out<<"Wavenumber(cm^-1)\tReal(Electrical)
            \tImag(Electrical)\tReal(Magnetic)\tImag(Magnetic)"<<endl;
    else
        PFC.Out<<"Wavenumber(cm^-1)\tElectrical\tMagnetic"<<endl;

    vector<double> _Wavenumber;
    vector<Wave> _Wave;
    vector<Intensity> Int;
    Wavenumber = PFC.LimitLower;

    while (Wavenumber <= PFC.LimitUpper) {

        FCM = FresnelCoefficient(DLayer, Wavenumber, Theta, Polarization);
        _Wavenumber.push_back(Wavenumber);

        if (FCoefficient.compare("rs") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.rs<<endl;
            _Wave.push_back(FCM.rs);
        }
    }
}

```

```

else if (FCoefficient.compare("rp") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.rp<<endl;
    _Wave.push_back(FCM.rp);
}

else if (FCoefficient.compare("ts") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.ts<<endl;
    _Wave.push_back(FCM.ts);
}

else if (FCoefficient.compare("tp") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.tp<<endl;
    _Wave.push_back(FCM.tp);
}

else if (FCoefficient.compare("Rs") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.Rs<<endl;
    Int.push_back(FCM.Rs);
}

else if (FCoefficient.compare("Rp") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.Rp<<endl;
    Int.push_back(FCM.Rp);
}

else if (FCoefficient.compare("Ts") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.Ts<<endl;
    Int.push_back(FCM.Ts);
}

else if (FCoefficient.compare("Tp") == 0) {
    PFC.Out<<Wavenumber<<"\t"<<FCM.Tp<<endl;
    Int.push_back(FCM.Tp);
}

Wavenumber += PFC.SizeStep;
}
PFC.Out.close();

if (_Wave.size() != 0)
    PFC.PlotWave(_Wavenumber, _Wave, FCoefficient);
else
    PFC.PlotIntensity(_Wavenumber, Int, FCoefficient);
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}

```

```

void DielectricLayer::
PlotFresnelCoefficient (DielectricLayer& DLayer, string FCoefficient, const double& Wavenumber,
                        Gnuplot& PFC)
{
    //Angles in degrees
    double Theta;
    FresnelCoefficientMatrix FCM;
    char Polarization;

    //Retrieving Polarization
    if ( FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 ||
        FCoefficient.compare("Rs") == 0 || FCoefficient.compare("Ts") == 0 ) {
        Polarization = 's';
    }

    else if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
        FCoefficient.compare("Rp") == 0 || FCoefficient.compare("Tp") == 0 ) {
        Polarization = 'p';
    }
    else {
        cout<<"\n\tUndefined Fresnel Coefficient"<<endl;
        exit(UNDEFINED_FRESNEL_COEF);
    }

    try {

        PFC.Out      <<"Fresnel Coefficient:"  <<FCoefficient<<endl
                    <<"Incident/Emerging"      <<endl
                    <<"Refractive Index of Ambient: "<<this->RefractiveIndex<<endl
                    <<DLayer <<endl<<endl
                    <<"Incident Light = "
                    <<Wavenumber<<" cm^-1 ("<<Polarization<<" Polarized)"
                    <<endl<<endl;

        if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
            FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 )

            PFC.Out<<"IncidentAngle(Deg)\tReal(Electrical)\tImag(Electrical)
                    \tReal(Magnetic)\tImag(Magnetic)"<<endl;
        else
            PFC.Out<<"IncidentAngle(Deg)\tElectrical\tMagnetic"<<endl;

        vector<double> _Theta;
        vector<Wave> _Wave;
        vector<Intensity> Int;

        Theta = PFC.LimitLower;
    }
}

```

```

while (Theta <= PFC.LimitUpper) {
    FCM = FresnelCoefficient(DLayer, Wavenumber, pi*Theta/180, Polarization);
    _Theta.push_back(Theta);

    if (FCoefficient.compare("rs") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.rs<<endl;
        _Wave.push_back(FCM.rs);
    }
    else if (FCoefficient.compare("rp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.rp<<endl;
        _Wave.push_back(FCM.rp);
    }
    else if (FCoefficient.compare("ts") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.ts<<endl;
        _Wave.push_back(FCM.ts);
    }
    else if (FCoefficient.compare("tp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.tp<<endl;
        _Wave.push_back(FCM.tp);
    }
    else if (FCoefficient.compare("Rs") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Rs<<endl;
        Int.push_back(FCM.Rs);
    }
    else if (FCoefficient.compare("Rp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Rp<<endl;
        Int.push_back(FCM.Rp);
    }
    else if (FCoefficient.compare("Ts") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Ts<<endl;
        Int.push_back(FCM.Ts);
    }
    else if (FCoefficient.compare("Tp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Tp<<endl;
        Int.push_back(FCM.Tp);
    }

    Theta += PFC.SizeStep;
}
PFC.Out.close();

if (_Wave.size() != 0)
    PFC.PlotWave(_Theta, _Wave, FCoefficient);
else
    PFC.PlotIntensity(_Theta, Int, FCoefficient);
}
catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}

```



```

void DielectricLayer::
PlotFresnelCoefficient (Ambient& AmbientMedium, string FCoefficient, Gnuplot& PFC,
                        const double& Theta)
{
    double Wavenumber;
    FresnelCoefficientMatrix FCM;
    char Polarization;

    //Retrieving Polarization
    if ( FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 ||
        FCoefficient.compare("Rs") == 0 || FCoefficient.compare("Ts") == 0 ) {
        Polarization = 's';
    }
    else if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
        FCoefficient.compare("Rp") == 0 || FCoefficient.compare("Tp") == 0 ) {
        Polarization = 'p';
    }
    else {
        cout<<"\n\tUndefined Fresnel Coefficient"<<endl;
        exit(UNDEFINED_FRESNEL_COEF);
    }

    try {
        PFC.Out      <<"Fresnel Coefficient:"  <<FCoefficient<<endl
                    <<"Incident/Emerging"    <<endl
                    <<"*this" <<endl
                    <<"Refractive Index of Ambient: "
                    <<AmbientMedium.GetRefractiveIndex()<<endl<<endl
                    <<"Incident Light = "<<Theta*180./pi<<" Deg ("
                    <<Polarization<<" Polarized)"<<endl<<endl;

        if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
            FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 )
            PFC.Out<<"Wavenumber(cm^-1)\tReal(Electrical)\t
                    Imag(Electrical)\tReal(Magnetic)\tImag(Magnetic)"<<endl;
        else
            PFC.Out<<"Wavenumber(cm^-1)\tElectrical\tMagnetic"<<endl;

        vector<double> _Wavenumber;
        vector<Wave> _Wave;
        vector<Intensity> Int;
        Wavenumber = PFC.LimitLower;

        while (Wavenumber <= PFC.LimitUpper) {

            FCM = FresnelCoefficient (AmbientMedium, Wavenumber, Theta,
                                    Polarization);
            _Wavenumber.push_back(Wavenumber);

```

```

        if (FCoefficient.compare("rs") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.rs<<endl;
            _Wave.push_back(FCM.rs);
        }
        else if (FCoefficient.compare("rp") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.rp<<endl;
            _Wave.push_back(FCM.rp);
        }
        else if (FCoefficient.compare("ts") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.ts<<endl;
            _Wave.push_back(FCM.ts);
        }
        else if (FCoefficient.compare("tp") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.tp<<endl;
            _Wave.push_back(FCM.tp);
        }
        else if (FCoefficient.compare("Rs") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.Rs<<endl;
            Int.push_back(FCM.Rs);
        }
        else if (FCoefficient.compare("Rp") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.Rp<<endl;
            Int.push_back(FCM.Rp);
        }
        else if (FCoefficient.compare("Ts") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.Ts<<endl;
            Int.push_back(FCM.Ts);
        }
        else if (FCoefficient.compare("Tp") == 0) {
            PFC.Out<<Wavenumber<<"\t"<<FCM.Tp<<endl;
            Int.push_back(FCM.Tp);
        }
        Wavenumber += PFC.SizeStep;
    }
    PFC.Out.close();

    if (_Wave.size() != 0)
        PFC.PlotWave(_Wavenumber, _Wave, FCoefficient);
    else
        PFC.PlotIntensity(_Wavenumber, Int, FCoefficient);
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}

```

```

void DielectricLayer::
PlotFresnelCoefficient (Ambient& AmbientMedium, string FCoefficient, const double& Wavenumber,
                        Gnuplot& PFC)
{
    double Theta;
    FresnelCoefficientMatrix FCM;
    char Polarization;

    //Retrieving Polarization
    if ( FCoefficient.compare("rs") == 0 ||
        FCoefficient.compare("ts") == 0 ||
        FCoefficient.compare("Rs") == 0 ||
        FCoefficient.compare("Ts") == 0 ) {
        Polarization = 's';
    }

    else if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
        FCoefficient.compare("Rp") == 0 || FCoefficient.compare("Tp") == 0 ) {
        Polarization = 'p';
    }

    else {
        cout<<"\n\tUndefined Fresnel Coefficient"<<endl;
        exit(UNDEFINED_FRESNEL_COEF);
    }

    try {

        PFC.Out      <<"Fresnel Coefficient:"  <<FCoefficient<<endl
                    <<"Incident/Emerging"    <<endl
                    <<"*this" <<endl
                    <<"Refractive Index of Ambient: "
                    <<AmbientMedium.GetRefractiveIndex()<<endl<<endl
                    <<"Incident Light = "
                    <<Wavenumber<<" cm^-1 ("<<Polarization<<" Polarized)"
                    <<endl<<endl;

        if ( FCoefficient.compare("rp") == 0 || FCoefficient.compare("tp") == 0 ||
            FCoefficient.compare("rs") == 0 || FCoefficient.compare("ts") == 0 )

            PFC.Out<<"IncidentAngle(Deg)\tReal(Electrical)\tImag(Electrical)\t
                    Real(Magnetic)\tImag(Magnetic)"<<endl;
        else
            PFC.Out<<"IncidentAngle(Deg)\tElectrical\tMagnetic"<<endl;

        vector<double> _Theta;
        vector<Wave> _Wave;
        vector<Intensity> Int;
    }
}

```

```

Theta = PFC.LimitLower;

while (Theta <= PFC.LimitUpper) {

    FCM = FresnelCoefficient(AmbientMedium, Wavenumber, Theta*pi/180.,
                             Polarization);
    _Theta.push_back(Theta);

    if (FCoefficient.compare("rs") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.rs<<endl;
        _Wave.push_back(FCM.rs);
    }
    else if (FCoefficient.compare("rp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.rp<<endl;
        _Wave.push_back(FCM.rp);
    }
    else if (FCoefficient.compare("ts") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.ts<<endl;
        _Wave.push_back(FCM.ts);
    }
    else if (FCoefficient.compare("tp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.tp<<endl;
        _Wave.push_back(FCM.tp);
    }
    else if (FCoefficient.compare("Rs") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Rs<<endl;
        Int.push_back(FCM.Rs);
    }
    else if (FCoefficient.compare("Rp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Rp<<endl;
        Int.push_back(FCM.Rp);
    }
    else if (FCoefficient.compare("Ts") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Ts<<endl;
        Int.push_back(FCM.Ts);
    }
    else if (FCoefficient.compare("Tp") == 0) {
        PFC.Out<<Theta<<"\t"<<FCM.Tp<<endl;
        Int.push_back(FCM.Tp);
    }
    Theta += PFC.SizeStep;
}
PFC.Out.close();

if (_Wave.size() != 0)
    PFC.PlotWave(_Theta, _Wave, FCoefficient);
else
    PFC.PlotIntensity(_Theta, Int, FCoefficient);
}

```

```

        catch (GnuplotException ge) {
            cout<<ge.what()<<endl;
        }
    }

//Used to Calculate the FCM for an interface between a DielectricLayer and Ambient medium
FresnelCoefficientMatrix
DielectricLayer::
FresnelCoefficient(DielectricLayer& DLayer, Ambient& IncidentMedium, const double& _Wavenumber,
const double& Theta, const char& Polarization)
{
    FresnelCoefficientMatrix FCM;

    COMPLEX n_In, mu_In, e_In;
    COMPLEX n_Out, mu_Out, e_Out;

    //Retrieving In Layer Parameters
    e_In = GetPermittivity(_Wavenumber);
    mu_In = GetPermeability(_Wavenumber);
    n_In = POW(e_In*mu_In - IncidentMedium.GetPermittivity()*
        IncidentMedium.GetPermeability()*pow(sin(Theta),2), 0.5);

    //Retrieving Out Layer Parameters
    e_Out = DLayer.GetPermittivity(_Wavenumber);
    mu_Out = DLayer.GetPermeability(_Wavenumber);
    n_Out = POW(e_Out*mu_Out - IncidentMedium.GetPermittivity()*
        IncidentMedium.GetPermeability()*pow(sin(Theta),2), 0.5);

    //Amplitudes and Intensity for s-Polarization
    if(Polarization == 's') {
        //Electrical Components for TE-Mode
        FCM.rs.Electrical = (n_In/mu_In - n_Out/mu_Out)/(n_In/mu_In + n_Out/mu_Out);
        FCM.ts.Electrical = (2.*n_In/mu_In)/(n_In/mu_In + n_Out/mu_Out);

        //Magnetic Components for TE-Mode
        FCM.rs.Magnetic = -FCM.rs.Electrical;
        FCM.ts.Magnetic = (pow(e_Out*mu_In/(e_In*mu_Out), 0.5))* FCM.ts.Electrical;

        //Optical Field for TE-Mode
        FCM.Rs.Magnitude = norm(FCM.rs.Electrical);
        FCM.Ts.Magnitude = norm(FCM.ts.Electrical) *
            real( (n_Out*mu_In)/(n_In*mu_Out) );
    }

    //Amplitudes and Intensity for p-Polarization
    else if (Polarization == 'p') {
        //Magnetic Components for for TM-Mode

```

```

FCM.rp.Magnetic = (n_In/e_In - n_Out/e_Out)/(n_In/e_In + n_Out/e_Out);
FCM.tp.Magnetic = (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

//Electrical Components for for TM-Mode
FCM.rp.Electrical = -FCM.rp.Magnetic;
FCM.tp.Electrical = pow(mu_Out*e_In/(mu_In*e_Out), 0.5)*(2.*n_In/e_In)/
(n_In/e_In + n_Out/e_Out);

//Optical Field for TE-Mode
FCM.Rp.Magnitude = norm(FCM.rp.Magnetic);
FCM.Tp.Magnitude = norm(FCM.tp.Magnetic)* real( (n_Out*e_In)/(n_In*e_Out) );
}

else {
    cerr<<"\n\tUndefined Polarization"<<endl;
    exit(UNDEFINED_POLARIZATION);
}
return FCM;
}

FresnelCoefficientMatrix DielectricLayer::
FresnelCoefficient(Ambient& EmergingMedium, Ambient& IncidentMedium,
    const double& _Wavenumber, const double& Theta, const char& Polarization)
{
    FresnelCoefficientMatrix FCM;

    COMPLEX n_In, mu_In, e_In;
    COMPLEX n_Out, mu_Out, e_Out;

    //Retrieving In Layer Parameters
    e_In = GetPermittivity(_Wavenumber);
    mu_In = GetPermeability(_Wavenumber);
    n_In = POW(e_In*mu_In - IncidentMedium.GetPermittivity()*
        IncidentMedium.GetPermeability()*pow(sin(Theta),2), 0.5);

    //Retrieving Out Layer Parameters
    e_Out = EmergingMedium.GetPermittivity();
    mu_Out = EmergingMedium.GetPermeability();
    n_Out = POW(e_Out*mu_Out - IncidentMedium.GetPermittivity()*
        IncidentMedium.GetPermeability()*pow(sin(Theta),2), 0.5);

    //Amplitudes and Intensity for s-Polarization
    if(Polarization == 's') {
        //Electrical Components for TE-Mode
        FCM.rs.Electrical = (n_In/mu_In - n_Out/mu_Out)/(n_In/mu_In + n_Out/mu_Out);
        FCM.ts.Electrical = (2.*n_In/mu_In)/(n_In/mu_In + n_Out/mu_Out);

        //Magnetic Components for TE-Mode
        FCM.rs.Magnetic = -FCM.rs.Electrical;
        FCM.ts.Magnetic = (pow(e_Out*mu_In/(e_In*mu_Out), 0.5))* FCM.ts.Electrical;
    }
}

```

```

//Optical Field for TE-Mode
FCM.Rs.Magnitude      = norm(FCM.rs.Electrical);
FCM.Ts.Magnitude      = norm(FCM.ts.Electrical) *
                        real( (n_Out*mu_In)/(n_In*mu_Out) );
}

//Amplitudes and Intensity for p-Polarization
else if (Polarization == 'p') {
    //Magnetic Components for for TM-Mode
    FCM.rp.Magnetic = (n_In/e_In - n_Out/e_Out)/(n_In/e_In + n_Out/e_Out);
    FCM.tp.Magnetic = (2.*n_In/e_In)/(n_In/e_In + n_Out/e_Out);

    //Electrical Components for for TM-Mode
    FCM.rp.Electrical = -FCM.rp.Magnetic;
    FCM.tp.Electrical = pow (mu_Out*e_In/(mu_In*e_Out), 0.5)*(2.*n_In/e_In)/
                        (n_In/e_In + n_Out/e_Out);

    //Optical Field for TE-Mode
    FCM.Rp.Magnitude = norm(FCM.rp.Magnetic);
    FCM.Tp.Magnitude = norm(FCM.tp.Magnetic)* real( (n_Out*e_In)/(n_In*e_Out) );
}

else {
    cerr<<"\n\tUndefined Polarization"<<endl;
    exit(UNDEFINED_POLARIZATION);
}
return FCM;
}

```

```

//Header for Class DielectricStack

#ifndef DielectricStack_H
#define DielectricStack_H

#include "DielectricLayer.h"
#include "Ambient.h"
#include "ThinFilmTransferMatrix.h"
#include "StokesVector.h"
#include "Gnuplot_i.h"
#include "NRUtil.h"

using namespace std;

typedef vector<double> vdouble;
typedef vector<vdouble> vvdouble;

//Forward Declarations of classes
class ThinFilmTransferMatrix;
class DielectricLayer;

void RemoveInterference(SpectralDataType&, SpectralDataType&, int, int, int);

const Ambient VACUUM(1.0, 1.0);

class DielectricStack {
protected:
    double  EpitaxialThickness;
    double  Thickness;
    double  EtchDepth;
    Ambient IncidentMedium;
    Ambient EmergingMedium;

    double  Temperature;
    double  IncidentAngle;
    string  IncidentSide;
    bool    IsSaved;

    //Used to check whether Ambients for the stack were already defined
    void    SetThickness(void);
    void    SetEpitaxialThickness(void);

public:
    vector<DielectricLayer> StackOfLayers;

    //Default Constructors
    DielectricStack(void);
    DielectricStack(string Message)
    :IncidentMedium(Message), EmergingMedium(Message){/*Neglect Message*/};

```



```

//Full Constructor
DielectricStack (const double& StackTemperature,
                 string SideIncident="Top or Bottom", const double& AngleIncident = 0.,
                 const Ambient& IncidentMedium = VACUUM,
                 const Ambient& EmergingMedium = VACUUM);

//Default Destructor
~DielectricStack(void){};

void SetLayer(int LayerIndex, const DielectricLayer& DLayer);
DielectricLayer GetLayer(int LayerIndex);

void    Read(void);
void    List(ostream&) const;
void    Save(void) const;
void    Seal(void);
void    Etch(double Depth = -1);
//Negative default will trigger user calls for incorrect input

//Incase the default constructor is called Initialization will have to be done immediately.
void    Initialize(const double& StackTemperature,
                 string SideIncident="Top or Bottom",
                 const double& AngleIncident = 0.,
                 const Ambient& IncidentMedium = VACUUM,
                 const Ambient& EmergingMedium = VACUUM);

bool    IsBulkTreatedCoherent;
void    SetTemperature(const double& StackTemperature);
double  GetTemperature(void) {return Temperature;}
double  GetThickness(void) {return Thickness;}
double  GetEpitaxialThickness(void) {return EpitaxialThickness;}
void    SetIncidentAngle(const double& AngleIncident)
        {IncidentAngle = AngleIncident;}
bool    SetIncidentSide(string Side="Top or Bottom");
string  GetIncidentSide(void){return IncidentSide;}
double  GetIncidentAngle(void){return IncidentAngle;}

//Generates the ThinFilmTransferMatrix for the stack
ThinFilmTransferMatrix GenerateTFTM
        (const double& Wavenumber, const char& Polarization);

COMPLEX ReflectionAmplitude      (double Wavenumber, char Polarization);
COMPLEX TransmissionAmplitude    (double Wavenumber, char Polarization);

//Returns single value for a specified Wavelength
//Polarized light
double Reflectance      (double Wavenumber, char Polarization);
double Transmittance    (double Wavenumber, char Polarization);

```

```
double Absorptance           (double Wavelength, char Polarization);
vector<double> AbsorptanceProbability (double Wavelength, char Polarization);
```

```
//Returns stokes vector for a specified Wavelength
//Unpolarized light
StokesVector Reflectance      (double Wavelength);
StokesVector Transmittance     (double Wavelength);
StokesVector Absorptance      (double Wavelength);
vector<double> AbsorptanceProbability (double Wavelength);
```

```
//Function calls for Angular and Wavelength Plots
//Variation of Components with Wavelength.
//Polarized Light
```

```
void PlotReflectanceRs      (Gnuplot& Plot);
void PlotReflectanceRp      (Gnuplot& Plot);
void PlotTransmittanceTs    (Gnuplot& Plot);
void PlotTransmittanceTp    (Gnuplot& Plot);
void PlotAbsorptanceAs      (Gnuplot& Plot);
void PlotAbsorptanceAp      (Gnuplot& Plot);
```

```
//Unpolarized Light
void PlotReflectance        (Gnuplot& Plot);
void PlotTransmittance      (Gnuplot& Plot);
void PlotAbsorptance        (Gnuplot& Plot);
```

```
//Variation of Components with Theta for a given Wavelength
```

```
//Polarized Light
void PlotReflectanceRs      (double Wavelength, Gnuplot& Plot);
void PlotReflectanceRp      (double Wavelength, Gnuplot& Plot);
void PlotTransmittanceTs    (double Wavelength, Gnuplot& Plot);
void PlotTransmittanceTp    (double Wavelength, Gnuplot& Plot);
void PlotAbsorptanceAs      (double Wavelength, Gnuplot& Plot);
void PlotAbsorptanceAp      (double Wavelength, Gnuplot& Plot);
```

```
//Unpolarized Light
void PlotReflectance        (double Wavelength, Gnuplot& Plot);
void PlotTransmittance      (double Wavelength, Gnuplot& Plot);
void PlotAbsorptance        (double Wavelength, Gnuplot& Plot);
```

```
//Absorptance Probability in individual layers
```

```
//Polarized Light
void PlotAbsorptanceProbability (Gnuplot& Plot, char Polarization);
void PlotAbsorptanceProbability (double Wavelength, char Polarization,
Gnuplot& Plot);
```

```

//Unpolarized Light
void PlotAbsorptanceProbability (Gnuplot& Plot);
void PlotAbsorptanceProbability (double Wavenumber, Gnuplot& Plot);

//Useful in sequential input of stacks.
inline friend istream& operator>> (istream& in, DielectricStack& DStack)
{
    DStack.Read();
    return in;
}

//Useful in sequential output of stacks.
inline friend ostream& operator<< (ostream& out, const DielectricStack& DStack)
{
    DStack.List(out);
    return out;
}

};

#endif

```

```

//Implementation of Dielectric Stack
#include "stdafx.h"
#include <fstream>
#include "DielectricStack.h"
#include "Read.h"

void ::RemoveInterference(SpectralDataType&, SpectralDataType&, int, int, int);

DielectricStack::DielectricStack(void)
{
    //Read in Stack details
    cin>>*this;

    if (StackOfLayers.size() == 0) {
        cout<<"\n\tEmpty Stack Found\n\t";
        exit(1);
    }
    else {
        cout<<*this;
        this->Save();
    }

    OutText("Enter the Temperature", true);
    cin>>Temperature;

    OutText("Enter the Incidence Angle", true);
    cin>>IncidentAngle;

    do {
        OutText("Enter the Incident Side", true);
        cin>>IncidentSide;
    } while (!SetIncidentSide(IncidentSide));

    //EtchDepth is assigned a negative value in order to activate user prompts in ::Etch(double)
    //when a depth value is not passed or an incorrect value is passed by user

    EtchDepth = -1.;
}

```

```

//Full Constructor
DielectricStack::
DielectricStack(const double& StackTemperature, string SideIncident, const double& AngleIncident,
                const Ambient& MediumIn, const Ambient& MediumOut)
    :IncidentMedium(MediumIn), EmergingMedium(MediumOut),
    IncidentAngle(AngleIncident), Temperature(StackTemperature),
    IncidentSide(SideIncident)
{
    //Order shouldnt be changed
    cin>>*this;

    if (StackOfLayers.size() == 0) {
        cout<<"\n\tEmpty Stack Found\n\t";
        exit(1);
    }

    else {
        cout<<*this;
        this->Save();
    }

    while (!SetIncidentSide(IncidentSide)) {
        OutText("Enter the Incident Side", true);
        cin>>IncidentSide;
    };

    EtchDepth = -1.;
}

//Used to Re-Initialize the Stack
void DielectricStack::
Initialize(const double& StackTemperature, string IncidentSide, const double& AngleIncident, const
Ambient& MediumIn, const Ambient& MediumOut)
{
    IncidentMedium      = MediumIn;
    EmergingMedium      = MediumOut;
    IncidentAngle       = AngleIncident;
    Temperature         = StackTemperature;

    while (!SetIncidentSide(IncidentSide)) {
        OutText("Enter the Incident Side", true);
        cin>>IncidentSide;
    };
}

```

```

ThinFilmTransferMatrix DielectricStack::
GenerateTFTM(const double& k, const char& Polarization)
{
    ThinFilmTransferMatrix TFTM;
    int i(0);
    COMPLEX n_Bulk;

    COMPLEX t_as=1., t_sa=1., r_as=0., r_sa=0.;
    COMPLEX t_bs, t_sb, r_bs, r_sb;
    COMPLEX t_ab, t_ba, r_ab, r_ba;
    COMPLEX a_Bulk =1.;

    //Evaluating the TFTM for IncidentMedium/FirstLayer
    TFTM *= TransferMatrix(IncidentMedium, StackOfLayers.at(i), k, IncidentAngle, Polarization);

    //Evaluating the TFTM for First through Last Layer of the Stack
    while(i<StackOfLayers.size()-1) {

        //Evaluating TFTM upto the point a "bulk layer" is encountered (for e.g. substrate)
        if (StackOfLayers.at(i+1).GetBulkOrFilm().compare("bulk") == 0) {

            //Barrier thickness will get set to its original value after the evaluation of TFTM
            double temp = StackOfLayers.at(i+1).GetThickness();
            StackOfLayers.at(i+1).SetThickness(0.);
            TFTM *= TransferMatrix(StackOfLayers.at(i), StackOfLayers.at(i+1),
                IncidentMedium, k, IncidentAngle, Polarization);
            StackOfLayers.at(i+1).SetThickness(temp);

            if (Polarization == 's') {
                t_as = 1./TFTM.s[0][0];
                t_sa = (TFTM.s[0][0]*TFTM.s[1][1] - TFTM.s[0][1]*TFTM.s[1][0])
                    /TFTM.s[0][0];
                r_as = TFTM.s[1][0]/TFTM.s[0][0];
                r_sa = -TFTM.s[0][1]/TFTM.s[0][0];
            }
            else if (Polarization == 'p') {
                t_as = 1./TFTM.p[0][0];
                t_sa = (TFTM.p[0][0]*TFTM.p[1][1] - TFTM.p[0][1]*TFTM.p[1][0])
                    /TFTM.p[0][0];
                r_as = TFTM.p[1][0]/TFTM.p[0][0];
                r_sa = -TFTM.p[0][1]/TFTM.p[0][0];
            }
        }

        //Evaluating the propogation inside the bulk material
        n_Bulk = POW(StackOfLayers.at(i+1).GetPermittivity(k)*
            StackOfLayers.at(i+1).GetPermeability(k) -
            IncidentMedium.GetPermittivity()*IncidentMedium.GetPermeability()
            *pow(sin(IncidentAngle),2), 0.5);

        a_Bulk = exp(I*2.*pi*k*n_Bulk*StackOfLayers.at(i+1).GetThickness());
    }
}

```

```

        /*Resetting TFTM will flush the matrix elements to evaluate the TFTM for next
        coherent part of the stack*/
        TFTM.Reset();
        i++;
    }

    //Evaluating TFTM for epitaxial layers that are not bulk
    else {
        TFTM *= TransferMatrix(StackOfLayers.at(i), StackOfLayers.at(i+1),
                               IncidentMedium, k, IncidentAngle, Polarization);
        i++;
    }
}

//Evaluating the TFTM for LastLayer/EmergingMedium Interface
TFTM *= TransferMatrix(StackOfLayers.at(i), EmergingMedium, IncidentMedium, k,
                       IncidentAngle, Polarization);

if (Polarization == 's') {
    t_sb = 1./TFTM.s[0][0];
    t_bs = (TFTM.s[0][0]*TFTM.s[1][1] - TFTM.s[0][1]*TFTM.s[1][0])/TFTM.s[0][0];
    r_sb = TFTM.s[1][0]/TFTM.s[0][0];
    r_bs = -TFTM.s[0][1]/TFTM.s[0][0];
}
else if (Polarization == 'p') {
    t_sb = 1./TFTM.p[0][0];
    t_bs = (TFTM.p[0][0]*TFTM.p[1][1] - TFTM.p[0][1]*TFTM.p[1][0])/TFTM.p[0][0];
    r_sb = TFTM.p[1][0]/TFTM.p[0][0];
    r_bs = -TFTM.p[0][1]/TFTM.p[0][0];
}

r_ab = r_as + t_as*t_sa*r_sb*pow(a_Bulk,2)/(1. - r_sa*r_sb*pow(a_Bulk,2));
t_ab = t_as*a_Bulk*t_sb/(1. - r_sa*r_sb*pow(a_Bulk,2));
r_ba = r_bs + t_bs*t_sb*r_sa*pow(a_Bulk,2)/(1. - r_sa*r_sb*pow(a_Bulk,2));
t_ba = t_bs*a_Bulk*t_sa/(1. - r_sa*r_sb*pow(a_Bulk,2));

if (Polarization == 's') {
    TFTM.s[0][0] = 1./t_ab;
    TFTM.s[0][1] = -r_ba/t_ab;
    TFTM.s[1][0] = r_ab/t_ab;
    TFTM.s[1][1] = (t_ab*t_ba - r_ab*r_ba)/t_ab;
}
else if (Polarization == 'p') {
    TFTM.p[0][0] = 1./t_ab;
    TFTM.p[0][1] = -r_ba/t_ab;
    TFTM.p[1][0] = r_ab/t_ab;
    TFTM.p[1][1] = (t_ab*t_ba - r_ab*r_ba)/t_ab;
}
}

```

```

//Returning "Coherent Amplitude" on user request
if (IsBulkTreatedCoherent == true)
    return TFTM;

//Evaluating TFTM coefficients for non Coherent "bulk material cases"
//This will smooth the spectral curves.
//This method is faster than the resolution method.

double R_ab(norm(r_as) + norm(t_as*t_sa*r_sb)*norm(pow(a_Bulk,2))
            /(1. - norm(r_sa*r_sb)*norm(pow(a_Bulk,2))));
double R_ba(norm(r_bs) + norm(t_bs*t_sb*r_sa)*norm(pow(a_Bulk,2))
            /(1. - norm(r_sb*r_sa)*norm(pow(a_Bulk,2))));
double T_ab(norm(t_as*t_sb)*norm(a_Bulk)/(1. - norm(r_sa*r_sb)*norm(pow(a_Bulk,2))));
double T_ba(norm(t_bs*t_sa)*norm(a_Bulk)/(1. - norm(r_sb*r_sa)*norm(pow(a_Bulk,2))));

if (Polarization == 's') {
    TFTM.s[0][0] = sqrt(1./T_ab);
    TFTM.s[0][1] = -sqrt(R_ba/T_ab);
    TFTM.s[1][0] = sqrt(R_ab/T_ab);
    TFTM.s[1][1] = sqrt((T_ab*T_ba - R_ab*R_ba)/T_ab);
}
else if (Polarization == 'p') {
    TFTM.p[0][0] = sqrt(1./T_ab);
    TFTM.p[0][1] = -sqrt(R_ba/T_ab);
    TFTM.p[1][0] = sqrt(R_ab/T_ab);
    TFTM.p[1][1] = sqrt((T_ab*T_ba - R_ab*R_ba)/T_ab);
}

return TFTM;
}

void DielectricStack::SetLayer(int LayerIndex, const DielectricLayer& DLayer)
{
    LayerIndex--;    //user input 1 ->Stack.Layer.at(0)

    if (LayerIndex<0 || LayerIndex>StackOfLayers.size()) {
        cout<<"\n\tLayer Index "<<LayerIndex+1<<" is invalid"<<endl;
        exit(INVALID_LAYER_IDX);
    }
    else
        StackOfLayers.at(LayerIndex) = DLayer;
}

```



```

DielectricLayer DielectricStack::GetLayer(int LayerIndex)
{
    LayerIndex--;    //user input 1 ->Stack.Layer.at(0)

    if (LayerIndex<0 || LayerIndex>StackOfLayers.size()) {
        cout<<"\n\tLayer Index "<<LayerIndex+1<<" is invalid"<<endl;
        exit(INVALID_LAYER_IDX);
    }
    else
        return StackOfLayers.at(LayerIndex);
}

void DielectricStack::Read(void)
{
    DielectricLayer DLayer;
    int Repeats, Singles, i, Temp;
    string Key;
    ifstream Infile;

    do Key = AskUser("Read structure from file? (y/n)", true, true);
    while (Key != "Y" && Key != "N");

    if (Key=="Y") {
        OpenFile(Infile);
        while(Infile>>DLayer) {
            StackOfLayers.push_back(DLayer);
            DLayer.Erase();
        }
        Infile.close();
    }
    else {
        IsSaved = true;
        cout<<"Parametrize layer structure from top to bottom\n";
        do{
            cout<<"Enter # of Repeats and Singles in a Repeat (eg: 10 2)\n";
            cin>>Repeats>>Singles;

            for (i=1;i<=Singles;i++) {
                cout<<"\nEnter data for layer "<<StackOfLayers.size()+1;
                cout<<"\nBulk/Film Material Thickness(m) DopantType(p,i,n)
                    Density(m^-3) Label AlloyFraction"<<endl;
                cin>>DLayer;
                StackOfLayers.push_back(DLayer);
                DLayer.Erase();
            }

            Temp = StackOfLayers.size()-Singles;

```

```

        for (i=Temp; i<Temp+Repeats*Singles; i++)
            StackOfLayers.push_back(StackOfLayers.at(i));
    } while (Repeats!=0 || Singles !=0);
}

//Thickness and Epitaxial Thickness should be set Immediately after the Read Call
SetThickness();
SetEpitaxialThickness();
}

void DielectricStack::List(ostream& out) const
{
    for (int i=0; i<StackOfLayers.size(); i++)
        out<<StackOfLayers.at(i)<<endl;
}

void DielectricStack::
Save(void) const
{
    string key;
    string FileName;
    ofstream Outfile;

    if (IsSaved == true)
        if (StackOfLayers.size() != 0) {
            do
                key = AskUser("Do you want to save the design? (y/n)", true, true);
            while (key != "Y" && key != "N");

            if (key == "Y") {
                FileName = AskUser("File name?", false, true);
                Outfile.open(FileName.c_str(), ios::out);
                for (int i=0; i<StackOfLayers.size(); i++)
                    Outfile<<StackOfLayers.at(i)<<endl;
            }
        }
}

void DielectricStack::SetThickness(void)
{
    Thickness = 0.;

    for (int i=0; i<StackOfLayers.size(); i++)
        Thickness += StackOfLayers.at(i).GetThickness();
}

```

```

void DielectricStack::SetEpitaxialThickness(void)
{
    EpitaxialThickness = 0.;

    for (int i=0; i<StackOfLayers.size(); i++)
        if (StackOfLayers.at(i).GetBulkOrFilm().compare("bulk") != 0)
            EpitaxialThickness += StackOfLayers.at(i).GetThickness();
}

void DielectricStack::Etch(double Depth)
{
    //if (EtchDepth < 1 Angstrom || or negative) do Nothing

    //if (undefined Stack) return(error)..
    if (StackOfLayers.size() == 0) {
        cerr<<"\tError! Stack is not Defined"<<endl;
        return;
    }

    EtchDepth = Depth;

    //Checking the Validity of EtchDepth
    while (1) {
        if (EtchDepth == -1) {
            cout<<"Enter Etch depth:\t";
            cin>>EtchDepth;
        }
        if (EtchDepth < 1e-8 )
            return;
        else if (EtchDepth >= EpitaxialThickness) {
            cout<<"\tInvalid Etch Depth"<<endl;
            EtchDepth = -1;
        }
        else break;
    }

    //do {etch} while (completed)
    do {
        if (StackOfLayers.at(0).GetBulkOrFilm().compare("bulk") == 0) {
            cout <<"Etching terminated at "<<StackOfLayers.at(0).GetBulkOrFilm()
                <<" "<<StackOfLayers.at(0).GetBaseMaterial()<<" interface"<<endl;
            return;
        }
        if (EtchDepth>=StackOfLayers.at(0).GetThickness()) {
            EtchDepth -= StackOfLayers.at(0).GetThickness();
            StackOfLayers.erase(StackOfLayers.begin());
        }
    }
}

```

```

        else {
            StackOfLayers.at(0).SetThickness(StackOfLayers.at(0).
            GetThickness()-EtchDepth);
            //Remove residual layers with thickness less than 1 Angstrom
            if (StackOfLayers.at(0).GetThickness() < 1.e-8)
                StackOfLayers.erase(StackOfLayers.begin());
            EtchDepth = 0.;
        }
        SetThickness();
        SetEpitaxialThickness();

    } while(EtchDepth >= 1e-8 );

    //Etching completed...
    EtchDepth = -1;
}

void DielectricStack::SetTemperature (const double& T)
{
    Temperature = T;

    for (int i=0; i<StackOfLayers.size(); i++)
        StackOfLayers.at(i).SetTemperature(T);
}

bool DielectricStack::SetIncidentSide(string Side)
{
    transform(Side.begin(), Side.end(), Side.begin(), tolower);

    if (string("top").compare(Side) !=0 && string("bottom").compare(Side) != 0 ) {
        cerr<<"\n\tIncident side "<<Side<<" is not defined"<<endl;
        return false;
    }
    else
        IncidentSide = Side;

    return true;
}

```



```

    else {
        if (IncidentSide.compare("top") == 0)
            return 1./TFTM.p[0][0];
        else
            return (TFTM.p[0][0]*TFTM.p[1][1]-TFTM.p[0][1]*
                    TFTM.p[1][0])/TFTM.p[0][0];
    }
    return 0.;
}

```

//Returns Polarized Components for a given Wavelength

```

double DielectricStack::Reflectance (double Wavenumber, char Polarization)
{
    return norm(ReflectionAmplitude(Wavenumber, Polarization));
}

```

```

double DielectricStack::Transmittance (double Wavenumber, char Polarization)
{
    COMPLEX e_In, e_Out, n_In, mu_In, n_Out, mu_Out;
    double Intensity;

    //Retrieving e,mu, and n from Layers
    e_In   = IncidentMedium.GetPermittivity();
    e_Out  = EmergingMedium.GetPermittivity();
    mu_In  = IncidentMedium.GetPermeability();
    mu_Out = EmergingMedium.GetPermeability();
    n_In   = POW(e_In*mu_In - pow(sin(IncidentAngle),2), 0.5);
    n_Out  = POW(e_Out*mu_Out - pow(sin(IncidentAngle),2), 0.5);

    Intensity = norm(TransmissionAmplitude(Wavenumber, Polarization));

    if (Polarization == 's')
        return Intensity * real((n_Out*mu_In)/(n_In*mu_Out));
    else
        return Intensity * real((n_Out*e_In)/(n_In*e_Out));
}

```

```

double DielectricStack::Absorptance (double Wavenumber, char Polarization)
{
    return 1- Reflectance(Wavenumber, Polarization) - Transmittance(Wavenumber, Polarization);
}

```

```

vector <double>
DielectricStack::AbsorptanceProbability (double Wavenumber, char Polarization)
{
    vdouble APvector;
    ThinFilmTransferMatrix TFTM;
    COMPLEX FieldForward, FieldReverse;

    COMPLEX r(ReflectionAmplitude(Wavenumber, Polarization));

    if (IncidentSide.compare("top") == 0) {

        for (int i=0; i<StackOfLayers.size(); i++) {
            //Evaluating the TFTM for IncidentMedium/FirstLayer
            COMPLEX n = StackOfLayers.at(i).GetRefractiveIndex();
            COMPLEX Dr = StackOfLayers.at(i).GetPermittivityForCarrierGeneration();

            double temp(StackOfLayers.at(i).GetThickness());
            StackOfLayers.at(i).SetThickness(0.);

            if (i==0)
                TFTM *= TransferMatrix(IncidentMedium, StackOfLayers.at(i),
                                         Wavenumber, IncidentAngle, Polarization);
            else
                TFTM *= TransferMatrix(StackOfLayers.at(i-1), StackOfLayers.at(i),
                                         IncidentMedium, Wavenumber, IncidentAngle, Polarization);

            if (Polarization == 's') {
                FieldForward = ( TFTM.s[1][1] - r*TFTM.s[0][1])/(TFTM.s[0][0]*
                                                                    TFTM.s[1][1] - TFTM.s[1][0]*TFTM.s[0][1]);
                FieldReverse = (-TFTM.s[1][0] + r*TFTM.s[0][0])/(TFTM.s[0][0]*
                                                                    TFTM.s[1][1] - TFTM.s[1][0]*TFTM.s[0][1]);
            }
            else if (Polarization == 'p') {
                FieldForward = ( TFTM.p[1][1] - r*TFTM.p[0][1])/(TFTM.p[0][0]*
                                                                    TFTM.p[1][1] - TFTM.p[1][0]*TFTM.p[0][1]);
                FieldReverse = (-TFTM.p[1][0] + r*TFTM.p[0][0])/(TFTM.p[0][0]*
                                                                    TFTM.p[1][1] - TFTM.p[1][0]*TFTM.p[0][1]);
            }

            StackOfLayers.at(i).SetThickness(temp);
            double z = pi*Wavenumber*StackOfLayers.at(i).GetThickness();
            //Phase at midpoint of the layer
            double J = norm(FieldForward*exp(I*n*z) + FieldReverse*exp(-I*n*z));
            //Intensity at midpoint of the layer
        }
    }
}

```

```

/*Measure to remove unwanted spiking caused by substrate
thickness*Wavenumber that exceed double limit*/

if (J*2*z*imag(Dr) > 10.)
    APvector.push_back(0.);
else
    APvector.push_back(J*2*z*imag(Dr));

TFTM *= TransferMatrix(StackOfLayers.at(i), StackOfLayers.at(i),
    IncidentMedium, Wavenumber, IncidentAngle, Polarization);

};
}
else if (IncidentSide.compare("bottom") == 0) {

    for (int i= StackOfLayers.size()-1; i>=0; i--) {
        //Evaluating the TFTM for IncidentMedium/FirstLayer
        COMPLEX n = StackOfLayers.at(i).GetRefractiveIndex();
        COMPLEX Dr = StackOfLayers.at(i).GetPermittivityForCarrierGeneration();

        double temp(StackOfLayers.at(i).GetThickness());
        StackOfLayers.at(i).SetThickness(0.);

        if (i==StackOfLayers.size()-1)
            TFTM *= TransferMatrix(EmergingMedium, StackOfLayers.at(i),
                Wavenumber, IncidentAngle, Polarization);
        else
            TFTM *= TransferMatrix(StackOfLayers.at(i+1), StackOfLayers.at(i),
                IncidentMedium, Wavenumber, IncidentAngle, Polarization);

        if (Polarization == 's') {
            FieldForward = ( TFTM.s[1][1] - r*TFTM.s[0][1])/(TFTM.s[0][0]*
                TFTM.s[1][1] - TFTM.s[1][0]*TFTM.s[0][1]);
            FieldReverse = (-TFTM.s[1][0] + r*TFTM.s[0][0])
                /(TFTM.s[0][0]*TFTM.s[1][1] - TFTM.s[1][0]*TFTM.s[0][1]);
        }
        else if (Polarization == 'p') {
            FieldForward = ( TFTM.p[1][1] - r*TFTM.p[0][1])/
                (TFTM.p[0][0]*TFTM.p[1][1] - TFTM.p[1][0]*TFTM.p[0][1]);
            FieldReverse = (-TFTM.p[1][0] + r*TFTM.p[0][0])/
                (TFTM.p[0][0]*TFTM.p[1][1] - TFTM.p[1][0]*TFTM.p[0][1]);
        }
        StackOfLayers.at(i).SetThickness(temp);
        double z = pi*Wavenumber*StackOfLayers.at(i).GetThickness();
        double J = norm(FieldForward*exp(I*n*z) + FieldReverse*exp(-I*n*z));

        if (J*2*z*imag(Dr) > 10.)
            APvector.push_back(0.);
        else
            APvector.push_back(J*2*z*imag(Dr));
    }
}

```



```

        TFTM *= TransferMatrix(StackOfLayers.at(i), StackOfLayers.at(i),
                                IncidentMedium, Wavenumber, IncidentAngle, Polarization);
    };
}

return APvector;
}
//-----End! Returns Polarized Components for a given Wavelength-----

//-----Start! Returns Components for unpolarized light at a given Wavelength-----

//Constructing Stokes components for unpolarized light
StokesVector DielectricStack::Reflectance (double Wavenumber)
{
    //Linear polarization basis
    COMPLEX Rs, Rp;
    StokesVector sVector;

    Rs = ReflectionAmplitude(Wavenumber, 's');
    Rp = ReflectionAmplitude(Wavenumber, 'p');

    sVector.s0 = 0.5*(norm(Rs) + norm(Rp));
    sVector.s1 = 0.5*(norm(Rs) - norm(Rp));
    sVector.s2 = real(Rs*conj(Rp));
    sVector.s3 = imag(Rs*conj(Rp));

    return sVector;
}

//Constructing Transmittance from Elliptical Components
StokesVector DielectricStack::Transmittance (double Wavenumber)
{
    //Linear polarization basis
    COMPLEX Ts, Tp;
    double sFactor, pFactor;
    StokesVector sVector;

    Ts = TransmissionAmplitude(Wavenumber, 's');
    Tp = TransmissionAmplitude(Wavenumber, 'p');

    COMPLEX e_In, e_Out, n_In, mu_In, n_Out, mu_Out;

    //Retrieving e,mu, and n from Layers
    e_In  = IncidentMedium.GetPermittivity();
    e_Out = EmergingMedium.GetPermittivity();

```

```

mu_In  = IncidentMedium.GetPermeability();
mu_Out = EmergingMedium.GetPermeability();
n_In   = POW(e_In*mu_In - pow(sin(IncidentAngle),2), 0.5);
n_Out  = POW(e_Out*mu_Out - pow(sin(IncidentAngle),2), 0.5);

sFactor = pow(real((n_Out*mu_In)/(n_In*mu_Out)), 0.5);
pFactor = pow(real((n_Out*e_In)/(n_In*e_Out)), 0.5);

sVector.s0 = 0.5*(norm(Ts*sFactor) + norm(Tp*pFactor));
sVector.s1 = 0.5*(norm(Ts*sFactor) - norm(Tp*pFactor));
sVector.s2 = real(Ts*sFactor*conj(Tp*pFactor));
sVector.s3 = imag(Ts*sFactor*conj(Tp*pFactor));

return sVector;
}

//Constructing Absorptance from Elliptical Components
StokesVector DielectricStack::Absorptance(double Wavenumber)
{
    StokesVector sVector;

    sVector.s0 = 1. - Reflectance(Wavenumber).s0 - Transmittance(Wavenumber).s0;
    sVector.s1 = 1. - Reflectance(Wavenumber).s1 - Transmittance(Wavenumber).s1;
    sVector.s2 = 1. - Reflectance(Wavenumber).s2 - Transmittance(Wavenumber).s2;
    sVector.s3 = 1. - Reflectance(Wavenumber).s3 - Transmittance(Wavenumber).s3;

    return sVector;;
}

vector<double>
DielectricStack::AbsorptanceProbability (double Wavenumber)
{
    vdouble AP_vector;

    vdouble AP_s_vector(AbsorptanceProbability(Wavenumber, 's'));
    vdouble AP_p_vector(AbsorptanceProbability(Wavenumber, 'p'));

    for (int i=0; i<StackOfLayers.size(); i++)
        AP_vector.push_back(0.5*(AP_s_vector.at(i) + AP_p_vector.at(i)));

    return AP_vector;
}

//-----End! Returns Components for unpolarized light at given Wavelength-----

```

```
//-----Start of Wavenumber Variation for Polarized Light-----
```

```
void DielectricStack::PlotReflectanceRs(Gnuplot& Plot)
{
    vector<double> _Wavenumber, Intensity;
    double dReflectance;

    try {

        Plot.Out      <<"Reflectance for Structure:\n"
                     <<*"this<<endl
                     <<"Incident Light = "
                     <<IncidentAngle*180./pi<<" Deg (s-Polarized)"
                     <<endl<<endl;

        Plot.Out      <<"TE Mode(s) Reflectance"<<endl<<endl;
        Plot.Out      <<"Wavenumber(cm^-1)\tReflectance(s)"<<endl;

        double Wavenumber = Plot.LimitLower;

        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            dReflectance = Reflectance(Wavenumber, 's');
            Plot.Out<<Wavenumber<<"\t"<<dReflectance<<endl;
            Intensity.push_back(dReflectance);
            Wavenumber += Plot.SizeStep;
        }

        Plot.Out.close();

        if (Intensity.size() != 0)
            Plot.plot_xy(_Wavenumber, Intensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}
```

```
void DielectricStack::PlotTransmittanceTs(Gnuplot& Plot)
{
    vector<double> _Wavenumber, Intensity;
    double dTransmittance;

    try {

        Plot.Out      <<"Transmittance for Structure:\n"
                     <<*"this<<endl
                     <<"Incident Light = "
```

```

        <<IncidentAngle*180./pi<<" Deg (s-Polarized)"
        <<endl<<endl;

    Plot.Out<<"TE Mode(s) Transmittance"<<endl<<endl;
    Plot.Out<<"Wavenumber(cm^-1)\tTransmittance(s)"<<endl;

    double Wavenumber = Plot.LimitLower;
    while (Wavenumber <= Plot.LimitUpper) {
        _Wavenumber.push_back(Wavenumber);
        dTransmittance = Transmittance(Wavenumber, 's');
        Plot.Out<<Wavenumber<<"\t"<<dTransmittance<<endl;
        Intensity.push_back(dTransmittance);
        Wavenumber += Plot.SizeStep;
    }

    Plot.Out.close();

    if (Intensity.size() != 0)
        Plot.plot_xy(_Wavenumber, Intensity, "h");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}

}

void DielectricStack::PlotReflectanceRp(Gnuplot& Plot)
{
    vector<double> _Wavenumber, Intensity;
    double dReflectance;

    try {
        Plot.Out        <<"Reflectance for Structure:\n"
                        <<*<<endl
                        <<"Incident Light = "
                        <<IncidentAngle*180./pi<<" Deg (p-Polarized)"
                        <<endl<<endl;

        Plot.Out        <<"TM Mode(p) Reflectance"<<endl<<endl;
        Plot.Out        <<"Wavenumber(cm^-1)\tReflectance(p)"<<endl;

        double Wavenumber = Plot.LimitLower;
        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            dReflectance = Reflectance(Wavenumber, 'p');
            Plot.Out<<Wavenumber<<"\t"<<dReflectance<<endl;
            Intensity.push_back(dReflectance);
            Wavenumber += Plot.SizeStep;
        }
    }
}

```

```

        Plot.Out.close();

        if (Intensity.size() != 0)
            Plot.plot_xy(_Wavenumber, Intensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

void DielectricStack::PlotTransmittanceTp(Gnuplot& Plot)
{
    vector<double> _Wavenumber, Intensity;
    double dTransmittance;

    try {
        Plot.Out        <<"Transmittance for Structure:\n"
                        <<*"this<<endl
                        <<"Incident Light = "
                        <<IncidentAngle*180./pi<<" Deg (p-Polarized)"
                        <<endl<<endl;

        Plot.Out        <<"TM Mode(p) Transmittance"<<endl<<endl;
        Plot.Out        <<"Wavenumber(cm^-1)\tTransmittance(p)"<<endl;

        double Wavenumber = Plot.LimitLower;
        while (Wavenumber <= Plot.LimitUpper) {
            dTransmittance = Transmittance(Wavenumber, 'p');
            _Wavenumber.push_back(Wavenumber);
            Plot.Out<<Wavenumber<<"\t"<<dTransmittance<<endl;
            Intensity.push_back(dTransmittance);
            Wavenumber += Plot.SizeStep;
        }
        Plot.Out.close();

        if (Intensity.size() != 0)
            Plot.plot_xy(_Wavenumber, Intensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

```

```

void DielectricStack::PlotAbsorptanceAs(Gnuplot& Plot)
{
    vector<double> _Wavenumber, Intensity;

    try {
        Plot.Out      <<"Absorptance for Structure:\n"
                      <<*<<endl
                      <<"Incident Light = "
                      <<IncidentAngle*180./pi<<" Deg (s-Polarized)"
                      <<endl<<endl;

        Plot.Out      <<"TE Mode(s) Absorptance"<<endl<<endl;
        Plot.Out      <<"Wavenumber(cm^-1)\tAbsorptance(s)"<<endl;

        double Wavenumber = Plot.LimitLower;
        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            Plot.Out<<Wavenumber<<"\t"<<Absorptance(Wavenumber, 's')<<endl;
            Intensity.push_back(Absorptance(Wavenumber, 's'));
            Wavenumber += Plot.SizeStep;
        }

        Plot.Out.close();

        if (Intensity.size() != 0)
            Plot.plot_xy(_Wavenumber, Intensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

```

```

void DielectricStack::PlotAbsorptanceAp(Gnuplot& Plot)
{
    vector<double> _Wavenumber, Intensity;

    try {
        Plot.Out      <<"Absorptance for Structure:\n"
                      <<*<<endl
                      <<"Incident Light = "
                      <<IncidentAngle*180./pi<<" Deg (p-Polarized)"
                      <<endl<<endl;

        Plot.Out      <<"TM Mode(p) Absorptance"<<endl<<endl;
        Plot.Out      <<"Wavenumber(cm^-1)\tAbsorptance(p)"<<endl;

        double Wavenumber = Plot.LimitLower;

```

```

while (Wavenumber <= Plot.LimitUpper) {
    _Wavenumber.push_back(Wavenumber);
    Plot.Out<<Wavenumber<<"\t"<<Absorptance(Wavenumber, 'p')<<endl;
    Intensity.push_back(Absorptance(Wavenumber, 'p'));
    Wavenumber += Plot.SizeStep;
}

Plot.Out.close();

if (Intensity.size() != 0)
    Plot.plot_xy(_Wavenumber, Intensity, "h");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}
//-----End of Wavenumber Variation for Polarized Light-----

//-----Start of Wavenumber Variation for Unpolarized Light-----

void DielectricStack::PlotReflectance(Gnuplot& Plot)
{
    vector<double> _Wavenumber;
    vector<StokesVector> SIntensity;
    StokesVector sVector;

    try {
        Plot.Out    <<"Reflectance:\n"
                    <<*this<<endl
                    <<"Incident Light = "
                    <<IncidentAngle*180./pi
                    <<endl<<endl;

        Plot.Out    <<"Reflectance"<<endl<<endl;
        Plot.Out    <<StokesVector::Info<<endl;
        Plot.Out    <<"Wavenumber(cm^-1)\tS0\tS1\tS2\tS3"<<endl;

        double Wavenumber = Plot.LimitLower;
        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            sVector = Reflectance(Wavenumber);
            Plot.Out<<Wavenumber<<"\t"<<sVector<<endl;
            SIntensity.push_back(sVector);
            Wavenumber += Plot.SizeStep;
        }

        Plot.Out.close();
    }
}

```

```

        if (SIntensity.size() != 0)
            Plot.PlotStokesVector(_Wavenumber, SIntensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

void DielectricStack::PlotTransmittance(Gnuplot& Plot)
{
    vector<double> _Wavenumber;
    vector<StokesVector> SIntensity;

    try {

        Plot.Out <<"Transmittance:\n"
                <<*this<<endl
                <<"Incident Light = "
                <<IncidentAngle*180./pi
                <<endl<<endl;

        Plot.Out <<"Transmittance"<<endl<<endl;
        Plot.Out <<StokesVector::Info<<endl;
        Plot.Out <<"Wavenumber(cm^-1)\tS0\tS1\tS2\tS3"<<endl;

        double Wavenumber = Plot.LimitLower;
        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            Plot.Out<<Wavenumber<<"\t"<<Transmittance(Wavenumber)<<endl;
            SIntensity.push_back(Transmittance(Wavenumber));
            Wavenumber += Plot.SizeStep;
        }

        Plot.Out.close();

        if (SIntensity.size() != 0)
            Plot.PlotStokesVector(_Wavenumber, SIntensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

```



```

void DielectricStack::PlotAbsorptance(Gnuplot& Plot)
{
    vector<double> _Wavenumber;
    vector<StokesVector> SIntensity;

    try {

        Plot.Out      <<"Absorptance:\n"
                     <<*this<<endl
                     <<"Incident Light = "
                     <<IncidentAngle*180./pi
                     <<endl<<endl;

        Plot.Out      <<"Absorptance"<<endl<<endl;
        Plot.Out      <<StokesVector::Info<<endl;
        Plot.Out      <<"Wavenumber(cm^-1)\tS0\tS1\tS2\tS3"<<endl;

        double Wavenumber = Plot.LimitLower;

        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            Plot.Out<<Wavenumber<<"\t"<<Absorptance(Wavenumber)<<endl;
            SIntensity.push_back(Absorptance(Wavenumber));
            Wavenumber += Plot.SizeStep;
        }

        Plot.Out.close();

        if (SIntensity.size() != 0)
            Plot.PlotStokesVector(_Wavenumber, SIntensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }
}

//-----End of Wavenumber Variation for Unpolarized Light-----

```

```
//-----Start of Angular Variation for Polarized Light-----
```

```
void DielectricStack::PlotReflectanceRs    (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta, Intensity;

    try {

        Plot.Out        <<"Reflectance for Structure:\n"
                        <<*<<endl
                        <<"Incident Light = "
                        <<"\nWavenumber\t"<<Wavenumber<<" (cm^-1)"
                        <<endl<<endl;

        Plot.Out        <<"TE Mode(s) Reflectance"<<endl<<endl;
        Plot.Out        <<"Theta (Deg)\tReflectance(s)"<<endl;

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"\t"<<Reflectance(Wavenumber, 's')<<endl;
            Intensity.push_back(Reflectance(Wavenumber, 's'));
            Theta += Plot.SizeStep;
        }

        Plot.Out.close();
        if (Intensity.size() != 0)
            Plot.plot_xy(_Theta, Intensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

void DielectricStack::PlotReflectanceRp    (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta, Intensity;

    try {

        Plot.Out        <<"Reflectance for Structure:\n"
                        <<*<<endl
                        <<"Incident Light = "
```

```

        <<"\nWavenumber\t"<<Wavenumber<<" (cm^-1)"
        <<endl<<endl;

    Plot.Out        <<"TM Mode(p) Reflectance"<<endl<<endl;
    Plot.Out        <<"Theta (Deg)\tReflectance(s)"<<endl;

    double Theta = Plot.LimitLower;
    while (Theta <= Plot.LimitUpper) {
        IncidentAngle = Theta*pi/180.;
        _Theta.push_back(Theta);
        Plot.Out<<Theta<<"\t"<<Reflectance(Wavenumber, 'p')<<endl;
        Intensity.push_back(Reflectance(Wavenumber, 'p'));
        Theta += Plot.SizeStep;
    }

    Plot.Out.close();

    if (Intensity.size() != 0)
        Plot.plot_xy(_Theta, Intensity, "h");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}

IncidentAngle = Buf;
}

void DielectricStack::PlotTransmittanceTs (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta, Intensity;

    try {

        Plot.Out        <<"Transmittance for Structure:\n"
        <<*this<<endl
        <<"Incident Light = "
        <<"\nWavenumber\t"<<Wavenumber<<" (cm^-1)"
        <<endl<<endl;

        Plot.Out        <<"TE Mode(s) Transmittance"<<endl<<endl;
        Plot.Out        <<"Theta (Deg)\tTransmittance(s)"<<endl;

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"\t"<<Transmittance(Wavenumber, 's')<<endl;

```

```

        Intensity.push_back(Transmittance(Wavenumber, 's'));
        Theta += Plot.SizeStep;
    }
    Plot.Out.close();

    if (Intensity.size() != 0)
        Plot.plot_xy(_Theta, Intensity, "h");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
IncidentAngle = Buf;
}

void DielectricStack::PlotTransmittanceTp (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta, Intensity;

    try {
        Plot.Out <<"Transmittance for Structure:\n"
        << *this<<endl
        <<"Incident Light = "
        <<"\nWavenumber\t"<<Wavenumber<<" (cm^-1)"
        <<endl<<endl;

        Plot.Out <<"TM Mode(p) Transmittance"<<endl<<endl;
        Plot.Out <<"Theta (Deg)\tTransmittance(s)"<<endl;

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"\t"<<Transmittance(Wavenumber, 'p')<<endl;
            Intensity.push_back(Transmittance(Wavenumber, 'p'));
            Theta += Plot.SizeStep;
        }
        Plot.Out.close();

        if (Intensity.size() != 0)
            Plot.plot_xy(_Theta, Intensity, "h");
    }
    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

```

```

void DielectricStack::PlotAbsorptanceAs (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta, Intensity;

    try {

        Plot.Out      <<"Absorptance for Structure:\n"
                     <<*this<<endl
                     <<"Incident Light = "
                     <<"\nWavenumber\t"<<Wavenumber<<" (cm^-1)"
                     <<endl<<endl;

        Plot.Out      <<"TE Mode(s) Absorptance"<<endl<<endl;
        Plot.Out      <<"Theta (Deg)\tAbsorptance(s)"<<endl;

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"\t"<<Absorptance(Wavenumber, 's')<<endl;
            Intensity.push_back(Absorptance(Wavenumber, 's'));
            Theta += Plot.SizeStep;
        }
        Plot.Out.close();

        if (Intensity.size() != 0)
            Plot.plot_xy(_Theta, Intensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

```

```

void DielectricStack::PlotAbsorptanceAp (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta, Intensity;

    try {

        Plot.Out      <<"Absorptance for Structure:\n"
                     <<*this<<endl
                     <<"Incident Light = "
                     <<"\nWavenumber\t"<<Wavenumber<<" (cm^-1)"
                     <<endl<<endl;

```

```

Plot.Out      <<"TM Mode(p) Absorptance"<<endl<<endl;
Plot.Out      <<"Theta (Deg)\tAbsorptance(p)"<<endl;

double Theta = Plot.LimitLower;
while (Theta <= Plot.LimitUpper) {
    IncidentAngle = Theta*pi/180.;
    _Theta.push_back(Theta);
    Plot.Out<<Theta<<"\t"<<Absorptance(Wavenumber, 'p')<<endl;
    Intensity.push_back(Absorptance(Wavenumber, 'p'));
    Theta += Plot.SizeStep;
}

Plot.Out.close();

if (Intensity.size() != 0)
    Plot.plot_xy(_Theta, Intensity, "h");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}

IncidentAngle = Buf;
}
//-----End of Angular Variation for Polarized Light-----

//-----Start of Angular Variation for Unpolarized Light-----

void DielectricStack::PlotReflectance(double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta;
    vector<StokesVector> SIntensity;

    try {
        Plot.Out      <<"Reflectance:\n"
                       <<*this<<endl
                       <<"Incident Light = "
                       <<"Wavenumber\t"<<Wavenumber<<" cm^-1"<<endl
                       <<endl<<endl;

        Plot.Out      <<"Reflectance"<<endl<<endl;
        Plot.Out      <<StokesVector::Info<<endl;
        Plot.Out      <<"Theta(Deg)\tS0\tS1\tS2\tS3"<<endl;
    }
}

```

```

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"t"<<Reflectance(Wavenumber)<<endl;
            SIntensity.push_back(Reflectance(Wavenumber));
            Theta += Plot.SizeStep;
        }
        Plot.Out.close();

        if (SIIntensity.size() != 0)
            Plot.PlotStokesVector(_Theta, SIIntensity, "h");
    }
    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

void DielectricStack::PlotTransmittance(double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta;
    vector<StokesVector> SIIntensity;

    try {
        Plot.Out <<"Transmittance:\n"
            <<*this<<endl
            <<"Incident Light = "
            <<"Wavenumber\t"<<Wavenumber<<" cm^-1"<<endl
            <<endl<<endl;

        Plot.Out <<"Transmittance"<<endl<<endl;
        Plot.Out <<StokesVector::Info<<endl;
        Plot.Out <<"Theta(Deg)\tS0\tS1\tS2\tS3"<<endl;

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"t"<<Transmittance(Wavenumber)<<endl;
            SIIntensity.push_back(Transmittance(Wavenumber));
            Theta += Plot.SizeStep;
        }

        Plot.Out.close();
    }
}

```

```

        if (SIntensity.size() != 0)
            Plot.PlotStokesVector(_Theta, SIntensity, "h");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

void DielectricStack::PlotAbsorptance(double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vector<double> _Theta;
    vector<StokesVector> SIntensity;

    try {
        Plot.Out      <<"Absorptance:\n"
                     <<*<<endl
                     <<"Incident Light = "
                     <<"Wavenumber\t"<<Wavenumber<<" cm^-1"<<endl
                     <<endl<<endl;

        Plot.Out      <<"Absorptance"<<endl<<endl;
        Plot.Out      <<StokesVector::Info<<endl;
        Plot.Out      <<"Theta(Deg)\tS0\tS1\tS2\tS3"<<endl;

        double Theta = Plot.LimitLower;
        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            Plot.Out<<Theta<<"\t"<<Absorptance(Wavenumber)<<endl;
            SIntensity.push_back(Absorptance(Wavenumber));
            Theta += Plot.SizeStep;
        }
        Plot.Out.close();

        if (SIntensity.size() != 0)
            Plot.PlotStokesVector(_Theta, SIntensity, "h");
    }
    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

//-----End of Angular Variation for Unpolarized Light-----

```



```

void DielectricStack::PlotAbsorptanceProbability(Gnuplot& Plot, char Polarization)
{
    vdouble _Wavenumber, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;

    bool Temp(!IsBulkTreatedCoherent);      //User call saved if digital filtering required
    IsBulkTreatedCoherent = true;
    // This will enforce evaluation of reflectance using a coherent substrate
    // Then the smoothing if called for by the user is done through the
    // "Savitzky-Golay digital Filtering"
    // This method ensures correct results for doped substrate.

    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out      <<"Absorptance Probability for Structure:\n"
                     <<"*this<<endl
                     <<"Incident Light = "
                     <<"IncidentAngle*180./pi<<" Deg ("<<Polarization<<"-Polarized)"
                     <<endl<<endl;

        if (Polarization == 's')
            Plot.Out <<"TE Mode(s) Absorptance Probability"<<endl<<endl;
        else
            Plot.Out <<"TM Mode(p) Absorptance Probability"<<endl<<endl;

        //Writing column heading of the file
        Plot.Out<<"Wavenumber(cm^-1)\t";
        for (int i=0; i<StackOfLayers.size(); i++)
            Plot.Out<<StackOfLayers.at(i).GetLabel()<<"\t";
        Plot.Out<<endl;

        //Filling Data
        double Wavenumber = Plot.LimitLower;

        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            vAbsorptanceProbability = AbsorptanceProbability(Wavenumber, Polarization);
            vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
            Wavenumber += Plot.SizeStep;
        }
    }
}

```

```

//Smoothing substrate interference on request
if (vvAbsorptanceProbability.size() != 0 && Temp){
    for (int i=0; i<StackOfLayers.size(); i++) {
        //Abstracting unsmoothed data from vvAbsorptanceProbability
        for (int j=0; j<_Wavenumber.size(); j++)
            UnsmoothedData.push_back(vvAbsorptanceProbability.at(j).at(i));

        //Moving window averaging method
        //Modify the parameters 10, 10, 4 to change smoothing intensity
        //The first two are the number of left and right data points if available,
        //and Third is order of polynom fitting the moving window of [10, 10]

        RemoveInterference(UnsmoothedData, SmoothedData, 10, 10, 4);

        //Data smoothed and stored for column i
        for (int k=0; k<SmoothedData.size(); k++)
            vvAbsorptanceProbability.at(k).at(i) = SmoothedData.at(k);

        while(!UnsmoothedData.empty())
            UnsmoothedData.pop_back();
    }
}

for (i=0; i<_Wavenumber.size(); i++) {
    Plot.Out<<"\n"<<_Wavenumber[i]<<"\t";
    for (int j=0; j<StackOfLayers.size(); j++)
        Plot.Out<<vvAbsorptanceProbability.at(i).at(j)<<"\t";
}
Plot.Out.close();

//Seperating out Absorptance probability
for (i=0; i<_Wavenumber.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {
        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
}

```

```

        _BottomContact.push_back(BottomContact);
        _Substrate.push_back(Substrate);
    }

    Plot.plot_xy(_Wavenumber, _TopContact, "Top Contact");
    Plot.plot_xy(_Wavenumber, _Emitter, "Emitter");
    Plot.plot_xy(_Wavenumber, _BottomContact, "Bottom Contact");
    Plot.plot_xy(_Wavenumber, _Substrate, "Substrate");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}

void DielectricStack::PlotAbsorptanceProbability(double Wavenumber, char Polarization, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vdouble _Theta, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;

    bool Temp(!IsBulkTreatedCoherent);
    IsBulkTreatedCoherent = true;

    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out <<"Absorptance Probability for Structure:\n"
            <<*this<<endl
            <<"Incident Light = "
            <<"Wavenumber\t"<<Wavenumber<<" cm^-1 ("<<Polarization<<"-Polarized)"
            <<endl<<endl;

        if (Polarization == 's')
            Plot.Out <<"TE Mode(s) Absorptance Probability"<<endl<<endl;
        else
            Plot.Out <<"TM Mode(p) Absorptance Probability"<<endl<<endl;

        //Writing column heading of the file
        Plot.Out<<"Theta(Deg)\t";
        for (int i=0; i<StackOfLayers.size(); i++)
            Plot.Out << StackOfLayers.at(i).GetLabel()<<"\t";
        Plot.Out << endl;
    }
}

```

```

//Filling Data
double Theta = Plot.LimitLower;

while (Theta <= Plot.LimitUpper) {
    IncidentAngle = Theta*pi/180.;
    _Theta.push_back(Theta);
    vAbsorptanceProbability = AbsorptanceProbability(Wavenumber, Polarization);
    vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
    Theta += Plot.SizeStep;
}

for (i=0; i<_Theta.size(); i++) {
    Plot.Out<<"n"<<_Theta[i]<<"t";
    for (int j=0; j<StackOfLayers.size(); j++)
        Plot.Out<<vvAbsorptanceProbability.at(i).at(j)<<"t";
}
Plot.Out.close();

//Seperating out Absorptance probability
for (i=0; i<_Theta.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {

        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

Plot.plot_xy(_Theta, _TopContact, "Top Contact");
Plot.plot_xy(_Theta, _Emitter, "Emitter");
Plot.plot_xy(_Theta, _BottomContact, "Bottom Contact");
Plot.plot_xy(_Theta, _Substrate, "Substrate");
}

```

```

        catch (GnuplotException ge) {
            cout<<ge.what()<<endl;
        }

        IncidentAngle = Buf;
    }

void DielectricStack::PlotAbsorptanceProbability    (Gnuplot& Plot)
{
    vdouble _Wavenumber, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vvdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;

    bool Temp(!IsBulkTreatedCoherent);
    IsBulkTreatedCoherent = true;
    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out <<"Absorptance Probability for Structure:\n"
            << *this<<endl
            <<"Incident Light = "
            <<IncidentAngle*180./pi<<" Deg"
            <<endl<<endl;

        //Writing column heading of the file
        Plot.Out << "Wavenumber(cm^-1)\t";

        for (int i=0; i<StackOfLayers.size(); i++)
            Plot.Out << StackOfLayers.at(i).GetLabel()<<"\t";
        Plot.Out << endl;

        //Filling Data
        double Wavenumber = Plot.LimitLower;

        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            vAbsorptanceProbability = AbsorptanceProbability(Wavenumber);
            vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
            Wavenumber += Plot.SizeStep;
        }

        if (vvAbsorptanceProbability.size() != 0 && Temp){
            for (int i=0; i<StackOfLayers.size(); i++) {
                for (int j=0; j<_Wavenumber.size(); j++)
                    UnsmoothedData.push_back(vvAbsorptanceProbability.at(j).at(i));
                RemoveInterference(UnsmoothedData, SmoothedData, 10, 10, 4);
            }
        }
    }
}

```

```

        //Data smoothed and stored for column i
        for (int k=0; k<SmoothedData.size(); k++)
            vvAbsorptanceProbability.at(k).at(i) = SmoothedData.at(k);

        while(!UnsmoothedData.empty())
            UnsmoothedData.pop_back();
    }

}

for (i=0; i<_Wavenumber.size(); i++) {
    Plot.Out<<"n"<<_Wavenumber[i]<<"t";
    for (int j=0; j<StackOfLayers.size(); j++)
        Plot.Out<<vvAbsorptanceProbability.at(i).at(j)<<"t";
    }
Plot.Out.close();

//Seperating out Absorptance probability
for (i=0; i<_Wavenumber.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {

        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);
        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);
        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

Plot.plot_xy(_Wavenumber, _TopContact, "Top Contact");
Plot.plot_xy(_Wavenumber, _Emitter, "Emitter");
Plot.plot_xy(_Wavenumber, _BottomContact, "Bottom Contact");
Plot.plot_xy(_Wavenumber, _Substrate, "Substrate");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}

}

```

```

void DielectricStack::PlotAbsorptanceProbability (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vdouble _Theta, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;
    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out << "Absorptance Probability for Structure:\n"
            << *this << endl
            << "Incident Light = "
            << "Wavenumber\t" << Wavenumber << " cm^-1 "
            << endl << endl;

        //Writing column heading of the file
        Plot.Out << "Theta(Deg)\t";
        for (int i=0; i<StackOfLayers.size(); i++)
            Plot.Out << StackOfLayers.at(i).GetLabel() << "\t";
        Plot.Out << endl;

        //Filling Data
        double Theta = Plot.LimitLower;

        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            vAbsorptanceProbability = AbsorptanceProbability(Wavenumber);
            vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
            Theta += Plot.SizeStep;
        }

        for (i=0; i<_Theta.size(); i++) {
            Plot.Out << "\n" << _Theta[i] << "\t";
            for (int j=0; j<StackOfLayers.size(); j++)
                Plot.Out << vvAbsorptanceProbability.at(i).at(j) << "\t";
        }
        Plot.Out.close();

        //Seperating out Absorptance probability
        for (i=0; i<_Theta.size(); i++) {
            double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

            for (int j=0; j<StackOfLayers.size(); j++) {

                if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
                    TopContact += vvAbsorptanceProbability.at(i).at(j);
            }
        }
    }
}

```

```

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

Plot.plot_xy(_Theta, _TopContact, "Top Contact");
Plot.plot_xy(_Theta, _Emitter, "Emitter");
Plot.plot_xy(_Theta, _BottomContact, "Bottom Contact");
Plot.plot_xy(_Theta, _Substrate, "Substrate");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}

IncidentAngle = Buf;
}

//Routines used by the /Remove_Interference/ call to smooth spectra
void lubksb(Mat_I_DP &a, Vec_I_INT &indx, Vec_IO_DP &b)
{
    int i,ii=0,ip,j; double sum;

    int n=a.nrows();
    for (i=0;i<n;i++) {
        ip=indx[i]; sum=b[ip]; b[ip]=b[i];
        if (ii != 0)
            for (j=ii-1;j<i;j++) sum -= a[i][j]*b[j];
        else if (sum != 0.0)
            ii=i+1;
        b[i]=sum;
    }
    for (i=n-1;i>=0;i--) {
        sum=b[i];
        for (j=i+1;j<n;j++) sum -= a[i][j]*b[j];
        b[i]=sum/a[i][i];
    }
}

```



```

void ludcmp(Mat_IO_DP &a, Vec_O_INT &indx, double &d)
{
    const double TINY=1.0e-20;
    int i,imax,j,k;
    double big,dum,sum,temp;

    int n=a.nrows();
    Vec_DP vv(n);
    d=1.0;
    for (i=0;i<n;i++) {
        big=0.0;
        for (j=0;j<n;j++)
            if ((temp=fabs(a[i][j])) > big) big=temp;
        if (big == 0.0) {
            cerr<<"Run-time error..."<<endl;
            cerr<<"Singular matrix in routine LU Decomposition routine\n";
            exit(11);
        }
        vv[i]=1.0/big;
    }

    for (j=0;j<n;j++) {
        for (i=0;i<j;i++) {
            sum=a[i][j];
            for (k=0;k<i;k++) sum -= a[i][k]*a[k][j];
            a[i][j]=sum;
        }
        big=0.0;
        for (i=j;i<n;i++) {
            sum=a[i][j];
            for (k=0;k<j;k++) sum -= a[i][k]*a[k][j];
            a[i][j]=sum;
            if ((dum=vv[i]*fabs(sum)) >= big) {
                big=dum;
                imax=i;
            }
        }
        if (j != imax) {
            for (k=0;k<n;k++) {
                dum=a[imax][k];
                a[imax][k]=a[j][k];
                a[j][k]=dum;
            }
            d = -d;
            vv[imax]=vv[j];
        }
        indx[j]=imax;
        if (a[j][j] == 0.0) a[j][j]=TINY;
        if (j != n-1) {

```

```

        dum=1.0/(a[j][j]);
        for (i=j+1;i<n;i++) a[i][j] *= dum;
    }
}

void savgol(Vec_O_DP &sgc, const int np, const int nl, const int nr, const int ld, const int m)
{
    int j, k, imj, ipj, kk, mm;
    double d, fac, sum;

    if(np < nl+nr+1 || nl < 0 || nr < 0 || ld>m || nl+nr < m) {
        cerr<<"Run-time error..."<<endl;
        cerr<<"Bad arguments encountered in Savitzky Golay Filtering\n";
        exit(10);
    }

    Vec_INT indx(m+1);
    Mat_DP a(m+1, m+1);
    Vec_DP b(m+1);

    //Setting up normal-equation matrices for the desired polynomial fit
    for(ipj=0; ipj<=(m<<1); ipj++) {
        sum = (ipj ? 0.0 : 1.0);
        for (k=1; k<=nr; k++)
            sum += pow(double(k), double(ipj));

        for (k=1; k<=nl; k++)
            sum += pow(double(-k), double(ipj));
        mm = MIN(ipj, 2*m-ipj);
        for (imj=-mm; imj<=mm; imj+=2)
            a[(ipj+imj)/2][(ipj-imj)/2] = sum;
    }

    //Solving for roots by LU decomposition
    ludcmp(a, indx, d);

    for (j=0; j<m+1; j++) b[j] = 0.;
    b[ld] = 1.;
    lubksb(a, indx, b);

    for(kk=0; kk<np-1; kk++) sgc[kk] = 0.;

    for(k=-nl; k<=nr; k++) {
        sum = b[0];
        fac = 1.0;

```

```

        for (mm=1; mm<=m; mm++)
            sum += b[mm]*(fac *= k);
        kk = (np-k) % np;
        sgc[kk] = sum;
    }
}

void RemoveInterference(SpectralDataType &UnsmoothedData, SpectralDataType &SmoothedData, int
NL, int NR, int M)
{
    int nL, nR, j;

    Vec_O_DP Sav_Gol_Coef(NL+NR+1);
    SmoothedData.resize(UnsmoothedData.size());

    for (int indx=0; indx<UnsmoothedData.size(); indx++) {
        SmoothedData.at(indx) = 0.;
        nL = NL;
        nR = NR;

        if (indx<nL)
            nL = indx;
        else if (indx >= UnsmoothedData.size()-nR)
            nR = UnsmoothedData.size()-indx-1;

        savgol(Sav_Gol_Coef,nL+nR+1, nL, nR, 0, M);

        for (j=0; j<=nL; j++)
            SmoothedData.at(indx) += Sav_Gol_Coef[j]*UnsmoothedData.at(indx-j);
        for (j=nL+1; j<nL+nR+1; j++)
            SmoothedData.at(indx) +=
Sav_Gol_Coef[j]*UnsmoothedData.at(nL+nR+1+indx-j);
    }
}

```

```

// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#ifndef AFX_STDAFX_H__FE46C2AD_E88C_447F_B2F4_AF04CA4C957F__INCLUDED_
#define AFX_STDAFX_H__FE46C2AD_E88C_447F_B2F4_AF04CA4C957F__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers

#include <stdio.h>

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_STDAFX_H__FE46C2AD_E88C_447F_B2F4_AF04CA4C957F__INCLUDED_)

// stdafx.cpp : source file that includes just the standard includes
// RespModel.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H and not in this file

```

```

//Header for Class InfraredDetector
//Derived as public from parent class "DielectricStack"

#ifndef INFRARED_DETECTOR_H
#define INFRARED_DETECTOR_H

#include "DielectricStack.h"

class InfraredDetector: public DielectricStack
{
    public:
        InfraredDetector(void){};
        InfraredDetector(string Message):DielectricStack(Message) {};

        //Full Constructor
        InfraredDetector(const double& StackTemperature,
                        string SideIncident="Top or Bottom",
                        const double& AngleIncident = 0.,
                        const Ambient& IncidentMedium = VACUUM,
                        const Ambient& EmergingMedium = VACUUM)
            :DielectricStack (StackTemperature, SideIncident, AngleIncident,
                            IncidentMedium, EmergingMedium) {};

        ~InfraredDetector(void){};

        vdouble Responsivity    (vdouble& _Wavenumber, vdouble& RespLayer,
                                const string Identifier, double _Thickness = 2.e-5,
                                double FermiLevel=14.29, double Workfunction=62.,
                                double HotColdScatLength = 4.e-5,
                                double HotPhononScatLength = 2.e-6);

                                //Lengths are in cm units to avoid conversions throughout
                                //The values assigned are only an example

        vdouble Responsivity    (double& _Wavenumber, vdouble& RespLayer,
                                const string Identifier, double _Thickness = 2.e-5,
                                double FermiLevel=14.29, double Workfunction=62.,
                                double HotColdScatLength = 4.e-5,
                                double HotPhononScatLength = 2.e-6);

        //Wavenumber Plot for given incidence angle
        void PlotResponsivity    (Gnuplot& Plot, char Polarization); //Polarized light
        void PlotResponsivity    (Gnuplot& Plot);                    //Unpolarized Light

        //Theta Plot for given Wavenumber
        void PlotResponsivity    (double Wavenumber, char Polarization, Gnuplot& Plot);
        void PlotResponsivity    (double Wavenumber, Gnuplot& Plot);

};
#endif;

```

```

//Implementation of class InfraredDetector
#include "stdafx.h"
#include "detector.h"

void InfraredDetector::PlotResponsivity (Gnuplot& Plot, char Polarization)
{
    vdouble _Wavenumber, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;
    int i;

    bool Temp(!IsBulkTreatedCoherent);
    IsBulkTreatedCoherent = true;
    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out <<"Responsivity for Structure:\n"
        <<*this<<endl
        <<"Incident Light = "
        <<IncidentAngle*180./pi<<" Deg ("<<Polarization<<"-Polarized)"
        <<endl<<endl;

        if (Polarization == 's')
            Plot.Out <<"TE Mode(s) Absorptance Probability"<<endl<<endl;
        else
            Plot.Out <<"TM Mode(p) Absorptance Probability"<<endl<<endl;

        //Writing column heading of the file
        Plot.Out <<"Wavenumber(cm^-1)\t"
        <<"Top Contact"<<"\t"
        <<"Emitter"<<"\t"
        <<"Bottom Contact"<<"\t"
        <<"Substrate"<<"\t";
        Plot.Out<<endl;

        //Filling Data
        double Wavenumber = Plot.LimitLower;

        while (Wavenumber <= Plot.LimitUpper) {
            _Wavenumber.push_back(Wavenumber);
            vAbsorptanceProbability = AbsorptanceProbability(Wavenumber, Polarization);
            vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
            Wavenumber += Plot.SizeStep;
        }
    }
}

```

```

//Smoothing substrate interference on request
if (vvAbsorptanceProbability.size() != 0 && Temp){
    for (int i=0; i<StackOfLayers.size(); i++) {
        //Abstracting unsmoothed data from vvAbsorptanceProbability
        for (int j=0; j<_Wavenumber.size(); j++)
            UnsmoothedData.push_back(vvAbsorptanceProbability.at(j).at(i));
        RemoveInterference(UnsmoothedData, SmoothedData, 10, 10, 4);

        for (int k=0; k<SmoothedData.size(); k++)
            vvAbsorptanceProbability.at(k).at(i) = SmoothedData.at(k);

        while(!UnsmoothedData.empty())
            UnsmoothedData.pop_back();
    }
}

//Seperating out Absorptance probability
for (i=0; i<_Wavenumber.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {
        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact") == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

//Evaluating individual components
vdouble Responsivity_TopContact(Responsivity
    (_Wavenumber, _TopContact, "Top Contact"));
vdouble Responsivity_Emitter
    (Responsivity(_Wavenumber, _Emitter, "Emitter", 2.e-6));
vdouble Responsivity_BottomContact
    (Responsivity(_Wavenumber, _BottomContact, "Bottom Contact", 7.e-5));

//Useful in doped substrate contact

```

```

vdouble Responsivity_Substrate
    (Responsivity(_Wavenumber, _Substrate, "Substrate",4.e-2));

//Writing Data to File
for (i=0; i<_Wavenumber.size(); i++) {
    Plot.Out<<"n"<<_Wavenumber[i]<<"\t"
        <<Responsivity_TopContact.at(i)<<"\t"
        <<Responsivity_Emitter.at(i)<<"\t"
        <<Responsivity_BottomContact.at(i)<<"\t";
        <<Responsivity_Substrate.at(i)<<"\t";
    }
    Plot.Out.close();

//Plotting individual components

    Plot.plot_xy(_Wavenumber, Responsivity_TopContact, "Top Contact");
    Plot.plot_xy(_Wavenumber, Responsivity_Emitter, "Emitter");
    Plot.plot_xy(_Wavenumber, Responsivity_BottomContact, "Bottom Contact");
    Plot.plot_xy(_Wavenumber, Responsivity_Substrate, "Substrate");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}

}

void InfraredDetector::PlotResponsivity    (Gnuplot& Plot)
{
    vdouble _Wavenumber, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;
    int i;

    bool Temp(!IsBulkTreatedCoherent);
    IsBulkTreatedCoherent = true;

    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out <<"Responsivity for Structure:\n"
            <<*this<<endl
            <<"Incident Light = "
            <<IncidentAngle*180./pi<<" Deg "
            <<endl<<endl;
    }
}

```



```

//Writing column heading of the file
Plot.Out <<"Wavenumber(cm^-1)\t"
      <<"Top Contact"<<"\t"
      <<"Emitter"<<"\t"
      <<"Bottom Contact"<<"\t"
      <<"Substrate"<<"\t";
Plot.Out <<endl;

double Wavenumber = Plot.LimitLower;

while (Wavenumber <= Plot.LimitUpper) {
    _Wavenumber.push_back(Wavenumber);

    vAbsorptanceProbability = AbsorptanceProbability(Wavenumber);
    vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
    Wavenumber += Plot.SizeStep;
}

//Smoothing substrate interference on request
if (vvAbsorptanceProbability.size() != 0 && Temp){
    for (int i=0; i<StackOfLayers.size(); i++) {
        //Abstracting unsmoothed data from vvAbsorptanceProbability
        for (int j=0; j<_Wavenumber.size(); j++)
            UnsmoothedData.push_back(vvAbsorptanceProbability.at(j).at(i));

        RemoveInterference(UnsmoothedData, SmoothedData, 10, 10, 4);

        for (int k=0; k<SmoothedData.size(); k++)
            vvAbsorptanceProbability.at(k).at(i) = SmoothedData.at(k);

        while(!UnsmoothedData.empty())
            UnsmoothedData.pop_back();
    }
}

//Seperating out Absorptance probability
for (i=0; i<_Wavenumber.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {

        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);
    }
}

```

```

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

//Evaluating individual components
vdouble Responsivity_TopContact
    (Responsivity(_Wavenumber, _TopContact, "Top Contact"));
vdouble Responsivity_Emitter
    (Responsivity(_Wavenumber, _Emitter, "Emitter", 2.e-6));
vdouble Responsivity_BottomContact
    (Responsivity(_Wavenumber, _BottomContact, "Bottom Contact", 7.e-5));

//Useful in doped substrate contact
vdouble Responsivity_Substrate
    (Responsivity(_Wavenumber, _Substrate, "Substrate", 4.e-2));

for (i=0; i<_Wavenumber.size(); i++) {
    Plot.Out <<"n"<<_Wavenumber[i]<<"\t"
        <<Responsivity_TopContact.at(i)<<"\t"
        <<Responsivity_Emitter.at(i)<<"\t"
        <<Responsivity_BottomContact.at(i)<<"\t";
        <<Responsivity_Substrate.at(i)<<"\t";
    }
    Plot.Out.close();

//Plotting individual components
Plot.plot_xy(_Wavenumber, Responsivity_TopContact, "Top Contact");
Plot.plot_xy(_Wavenumber, Responsivity_Emitter, "Emitter");
Plot.plot_xy(_Wavenumber, Responsivity_BottomContact, "Bottom Contact");
Plot.plot_xy(_Wavenumber, Responsivity_Substrate, "Substrate");
}

catch (GnuplotException ge) {
    cout<<ge.what()<<endl;
}
}

```

```

void InfraredDetector::PlotResponsivity (double Wavenumber, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vdouble _Theta, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;

    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out <<"Resposivity for Structure:\n"
            <<*"this"<<endl
            <<"Incident Light = "
            <<"Wavenumber\t"<<Wavenumber<<" cm^-1 "
            <<endl<<endl;

        Plot.Out <<"Theta(Deg)\t"
            <<"Top Contact"<<"\t"
            <<"Emitter"<<"\t"
            <<"Bottom Contact"<<"\t"
            <<"Substrate"<<"\t";
        Plot.Out <<endl;

        //Filling Data
        double Theta = Plot.LimitLower;

        while (Theta <= Plot.LimitUpper) {
            IncidentAngle = Theta*pi/180.;
            _Theta.push_back(Theta);
            vAbsorptanceProbability = AbsorptanceProbability(Wavenumber);
            vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
            Theta += Plot.SizeStep;
        }

        //Seperating out Absorptance probability
        for (int i=0; i<_Theta.size(); i++) {
            double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

            for (int j=0; j<StackOfLayers.size(); j++) {

                if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
                    TopContact += vvAbsorptanceProbability.at(i).at(j);

                else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
                    BottomContact += vvAbsorptanceProbability.at(i).at(j);

                else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
                    Emitter += vvAbsorptanceProbability.at(i).at(j);
            }
        }
    }
}

```

```

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

//Seperating out Absorptance probability
for (i=0; i<_Theta.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {

        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

vdouble Responsivity_TopContact
    (Responsivity(Wavenumber, _TopContact, "Top Contact"));
vdouble Responsivity_Emitter
    (Responsivity(Wavenumber, _Emitter, "Emitter", 2.e-6));
vdouble Responsivity_BottomContact
    (Responsivity(Wavenumber, _BottomContact, "Bottom Contact", 7.e-5));

//Useful in doped substrate contact
vdouble Responsivity_Substrate
    (Responsivity(Wavenumber, _Substrate, "Substrate", 4.e-2));

//Writing Data to File

```

```

        for (i=0; i<_Theta.size(); i++) {
            Plot.Out <<"n"<<_Theta[i]<<"t"
                <<Responsivity_TopContact.at(i)<<"t"
                <<Responsivity_Emitter.at(i)<<"t"
                <<Responsivity_BottomContact.at(i)<<"t";
                <<Responsivity_Substrate.at(i)<<"t";
        }
        Plot.Out.close();

        Plot.plot_xy(_Theta, Responsivity_TopContact, "Top Contact");
        Plot.plot_xy(_Theta, Responsivity_Emitter, "Emitter");
        Plot.plot_xy(_Theta, Responsivity_BottomContact, "Bottom Contact");
        Plot.plot_xy(_Theta, Responsivity_Substrate, "Substrate");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

void InfraredDetector::PlotResponsivity    (double Wavenumber, char Polarization, Gnuplot& Plot)
{
    double Buf(IncidentAngle);
    vdouble _Theta, vAbsorptanceProbability;
    vdouble _TopContact, _Emitter, _BottomContact, _Substrate;
    vdouble vvAbsorptanceProbability;
    SpectralDataType UnsmoothedData, SmoothedData;

    vAbsorptanceProbability.resize(StackOfLayers.size());

    try {
        //Details saved in the file
        Plot.Out <<"Respoinsivity for Structure:\n"
            <<*<<endl
            <<"Incident Light = "
            <<"Wavenumber\t"<<Wavenumber<<" cm^-1 ("<<Polarization<<"-Polarized)"
            <<endl<<endl;

        if (Polarization == 's')
            Plot.Out <<"TE Mode(s) Absorptance Probability"<<endl<<endl;
        else
            Plot.Out <<"TM Mode(p) Absorptance Probability"<<endl<<endl;
    }
}

```

```

Plot.Out <<"Theta(Deg)\t"
    <<"Top Contact"<<"\t"
    <<"Emitter"<<"\t"
    <<"Bottom Contact"<<"\t"
    <<"Substrate"<<"\t";
Plot.Out <<endl;

//Filling Data
double Theta = Plot.LimitLower;

while (Theta <= Plot.LimitUpper) {
    IncidentAngle = Theta*pi/180.;
    _Theta.push_back(Theta);
    vAbsorptanceProbability = AbsorptanceProbability(Wavenumber, Polarization);
    vvAbsorptanceProbability.push_back(vAbsorptanceProbability);
    Theta += Plot.SizeStep;
}

//Seperating out Absorptance probability
for (int i=0; i<_Theta.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {

        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact" ) == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

```

```

//Seperating out Absorptance probability
for (i=0; i<_Theta.size(); i++) {
    double TopContact(0.), Emitter(0.), BottomContact(0.), Substrate(0.);

    for (int j=0; j<StackOfLayers.size(); j++) {

        if(StackOfLayers.at(j).GetLabel().compare("topcontact") == 0 )
            TopContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("bottomcontact") == 0 )
            BottomContact += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("emitter") == 0 )
            Emitter += vvAbsorptanceProbability.at(i).at(j);

        else if (StackOfLayers.at(j).GetLabel().compare("substrate") == 0 )
            Substrate += vvAbsorptanceProbability.at(i).at(j);
    }

    _TopContact.push_back(TopContact);
    _Emitter.push_back(Emitter);
    _BottomContact.push_back(BottomContact);
    _Substrate.push_back(Substrate);
}

//Evaluating individual components
vdouble Responsivity_TopContact
    (Responsivity(Wavenumber, _TopContact, "Top Contact"));
vdouble Responsivity_Emitter
    (Responsivity(Wavenumber, _Emitter, "Emitter", 2.e-6));
vdouble Responsivity_BottomContact
    (Responsivity(Wavenumber, _BottomContact, "Bottom Contact", 7.e-5));

//Useful in doped substrate contact
vdouble Responsivity_Substrate
    (Responsivity(Wavenumber, _Substrate, "Substrate", 4.e-2));

//Writing Data to File
for (i=0; i<_Theta.size(); i++) {
    Plot.Out <<"\n"<<_Theta[i]<<"\t"
        <<Responsivity_TopContact.at(i)<<"\t"
        <<Responsivity_Emitter.at(i)<<"\t"
        <<Responsivity_BottomContact.at(i)<<"\t";
        <<Responsivity_Substrate.at(i)<<"\t";
}
Plot.Out.close();

```

```

        Plot.plot_xy(_Theta, Responsivity_TopContact, "Top Contact");
        Plot.plot_xy(_Theta, Responsivity_Emitter, "Emitter");
        Plot.plot_xy(_Theta, Responsivity_BottomContact, "Bottom Contact");
        Plot.plot_xy(_Theta, Responsivity_Substrate, "Substrate");
    }

    catch (GnuplotException ge) {
        cout<<ge.what()<<endl;
    }

    IncidentAngle = Buf;
}

vdouble InfraredDetector::
Responsivity (vdouble& _Wavenumber, vdouble& RespLayer, const string Identifier, double w,
              double FermiLevel, double Workfunction, double Le, double Lp)
{
    vdouble _Responsivity;

    if (w == 0.) {
        OutText("Pls. input Workfunction in meV", 0);
        cin>>w;
    }

    if (Workfunction == 0.) {
        OutText("Pls. input Workfunction in meV", 0);
        cin>>Workfunction;
    }

    if (FermiLevel == 0.) {
        OutText("Pls. input FermiLevel in meV", 0);
        cin>>FermiLevel;
    }

    if (Le == 0.) {
        OutText("Pls. input Hot carrier - Cold carrier scattering length in Angstrom", 0);
        cin>>Le;
    }

    if (Lp == 0.) {
        OutText("Pls. input Hot carrier - Phonon scattering length in Angstrom", 0);
        cin>>Lp;
    }

    //Fraction of Electron captured prior to Bulk Scattering
    double L(Le*Lp/(Le+Lp)), G(Le/(Le+Lp));
    double Ideal_Eta;
    double Eta_0 = (L/w)*pow(1-exp(-w/L), 0.5), EtaM;

```



```

for (int i=0; i<_Wavenumber.size(); i++) {
    double E = 0.124*_Wavenumber.at(i);

    if (Workfunction<FermiLevel)
    {
        if (E<Workfunction)
            Ideal_Eta=0;

        if ((Workfunction<=E) && (E<FermiLevel)) {
            Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                pow( FermiLevel+Workfunction,1.5)) -
                (E-Workfunction)*pow(FermiLevel+Workfunction, 0.5))
                /(pow( FermiLevel+E,1.5)-pow( FermiLevel,1.5)));

            EtaM = (pow(FermiLevel+E,1.5) –
                pow( FermiLevel+Workfunction,1.5))
                /(pow( FermiLevel+E,1.5)-pow( FermiLevel,1.5));
        }

        if ((E >=FermiLevel) && (E<FermiLevel+Workfunction))
        {
            Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                pow( FermiLevel+Workfunction, 1.5)) -
                (E-Workfunction)*pow(FermiLevel+Workfunction, 0.5))
                /(pow( FermiLevel+E,1.5)-pow( E,1.5)));

            EtaM = (pow(FermiLevel+E,1.5) –
                pow( FermiLevel+Workfunction,1.5))/
                (pow( FermiLevel+E,1.5)-pow( E,1.5));
        }

        if (E>=FermiLevel+Workfunction)
        {
            Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                pow( E, 1.5)) - (FermiLevel)*
                pow(FermiLevel+Workfunction, 0.5))
                /(pow( FermiLevel+E,1.5)-pow( E,1.5)));
            EtaM = 1;
        }
    }

    else if (FermiLevel<Workfunction)
    {
        if ((Workfunction<E) && (E<(FermiLevel+Workfunction)))
        {
            Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                pow( FermiLevel+Workfunction,1.5)) -
                (E-Workfunction)*pow(FermiLevel+Workfunction, 0.5)) /
                (pow( FermiLevel+E,1.5)-pow( E,1.5)));
        }
    }
}

```

```

        EtaM = (pow(FermiLevel+E,1.5) -
                pow( FermiLevel+Workfunction,1.5))/
                (pow( FermiLevel+E,1.5)-pow( E,1.5));
    }
    if (E > (FermiLevel + Workfunction))
    {
        Ideal_Eta = (3/4.0) *
                    (((2/3.0)*(pow(FermiLevel+E,1.5) - pow( E, 1.5)) -
                      FermiLevel*pow(FermiLevel+Workfunction, 0.5)) /
                     (pow( FermiLevel+E,1.5)-pow( E,1.5))));
        EtaM = 1.0;
    }

    if (E<Workfunction)
        Ideal_Eta = 0;
}

_Responsevity.push_back
    (RespLayer.at(i)*Eta_0*Ideal_Eta/(1-G+(G*Ideal_Eta*Eta_0/EtaM)));
}
return _Responsivity;
}

```

vdouble InfraredDetector::

Responsivity(double& Wavenumber, vdouble& RespLayer, const string Identifier, double w,
double FermiLevel, double Workfunction, double Le, double Lp)

```

{
    vdouble _Responsivity;

    if (w == 0.) {
        OutText("Pls. input Workfunction in meV", 0);
        cin>>w;
    }

    if (Workfunction == 0.) {
        OutText("Pls. input Workfunction in meV", 0);
        cin>>Workfunction;
    }

    if (FermiLevel == 0.) {
        OutText("Pls. input FermiLevel in meV", 0);
        cin>>FermiLevel;
    }

    if (Le == 0.) {
        OutText("Pls. input Hot carrier - Cold carrier scattering length in Angstrom", 0);
        cin>>Le;
    }
}

```

```

if (Lp == 0.) {
    OutText("Pls. input Hot carrier - Phonon scattering length in Angstrom", 0);
    cin>>Lp;
}

//Fraction of Electron captured prior to Bulk Scattering
double L(Le*Lp/(Le+Lp)), G(Le/(Le+Lp));
double Ideal_Eta;
double Eta_0 = (L/w)*pow(1-exp(-w/L), 0.5), EtaM;

for (int i=0; i<RespLayer.size(); i++) {

    double E = 0.124*Wavenumber;

    if (E<Workfunction) _Responsivity.push_back(0.);

    else {
        if (Workfunction<FermiLevel) {
            if (E<Workfunction) Ideal_Eta=0;

            if ((Workfunction<=E) && (E<FermiLevel)) {
                Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                    pow( FermiLevel+Workfunction,1.5)) -
                    (E-Workfunction)*pow(FermiLevel+Workfunction, 0.5))
                    /(pow( FermiLevel+E,1.5)-pow( FermiLevel,1.5)));
                EtaM = (pow(FermiLevel+E,1.5) –
                    pow( FermiLevel+Workfunction,1.5))/
                    (pow( FermiLevel+E,1.5)-pow( FermiLevel,1.5));
            }

            if ((E >=FermiLevel) && (E<FermiLevel+Workfunction)) {
                Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                    pow( FermiLevel+Workfunction, 1.5)) -
                    (E-Workfunction)*pow(FermiLevel+Workfunction, 0.5)) /
                    (pow( FermiLevel+E,1.5)-pow( E,1.5)));

                EtaM = (pow(FermiLevel+E,1.5) –
                    pow( FermiLevel+Workfunction,1.5))/
                    (pow( FermiLevel+E,1.5)-pow( E,1.5));
            }

            if (E>=FermiLevel+Workfunction) {
                Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) –
                    pow( E, 1.5)) - (FermiLevel)*pow(FermiLevel+
                    Workfunction, 0.5)) /(pow( FermiLevel+E,1.5)-
                    pow( E,1.5)));
                EtaM = 1;
            }
        }
    }
}

```

```

else if (FermiLevel<Workfunction) {
    if ((Workfunction<E) && (E<(FermiLevel+Workfunction))) {
        Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) -
            pow( FermiLevel+Workfunction,1.5)) -
            (E-Workfunction)*pow(FermiLevel+Workfunction, 0.5)) /
            (pow( FermiLevel+E,1.5)-pow( E,1.5)));
        EtaM = (pow(FermiLevel+E,1.5) -
            pow( FermiLevel+Workfunction,1.5))
            /(pow( FermiLevel+E,1.5)-pow( E,1.5));
    }

    if (E > (FermiLevel + Workfunction)) {
        Ideal_Eta = (3/4.0) * (((2/3.0)*(pow(FermiLevel+E,1.5) -
            pow( E, 1.5)) - FermiLevel*
            pow(FermiLevel+Workfunction, 0.5)) /
            (pow( FermiLevel+E,1.5)-pow( E,1.5)));
        EtaM = 1.0;
    }

    if (E<Workfunction)
        Ideal_Eta = 0;
}

}

_Responseivity.push_back
(RespLayer.at(i)*Eta_0*Ideal_Eta/(1-G+(G*Ideal_Eta*Eta_0/EtaM)));
}
return _Responsivity;
}

```

/*RespModel is used as a white board to define dielectric stacks and detectors and call their various member functions. Following call routines are few examples only! */

```
#include "stdafx.h"
#include "DielectricLayer.h"
#include "Ambient.h"
#include "DielectricStack.h"
#include "Detector.h"
#include "OpticalConstants.h"
#include <sstream>

int main(int argc, char* argv[])
{
    InfraredDetector HE01(4.2, "top");
    Ambient Vacuum(VACUUM);
    HE01.IsBulkTreatedCoherent = false;

    OpticalConstants OC;
    OC.SetDefaultPath("c:/Optical Constants Library/");
    OC.SetNonDispersiveConstantsForStack(HE01);

    HE01.SetIncidentAngle(0.*pi/180.);

    Gnuplot g1 = Gnuplot("lines");
    g1.InitializePlot("HE01Resp1.dat", 90, 0, 0.1, "Theta(Deg)", "Responsivity");
    HE01.PlotResponsivity(1000,'p', g1);
    getch();

    HE01.SetIncidentAngle(45.*pi/180.);

    Gnuplot g2 = Gnuplot("lines");
    g2.InitializePlot("HE01Resp2.dat", 2000, 50, 1, "Wavenumber (cm^-1)", "A_10");
    HE01.PlotResponsivity(g2);
    getch();

    DielectricStack Structure(4.2, "top");
    Ambient Vacuum(VACUUM);
    Structure.IsBulkTreatedCoherent = false;

    OpticalConstants OC;
    OC.SetDefaultPath("c:/Optical Constants Library/");
    OC.SetNonDispersiveConstantsForStack(Structure);

    Structure.SetIncidentAngle(0.*pi/180.);

    Gnuplot g1 = Gnuplot("lines");
    g1.InitializePlot("test.dat", 2000, 50, 1, "Wavenumber (cm^-1)", "A_10");
    Structure.PlotReflectance(g1);
```

```

cout<<Structure.AbsorptanceProbability(50, 's');
getch();

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("RsSmoothed.dat", 2000, 50, 1, "Theta", "Reflectance, Rs");
Structure.PlotAbsorptanceProbability(g1, 's');

Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("RsUnSmoothed.dat", 2000, 100, 1, "Theta", "Reflectance, Rs");
Structure.PlotReflectanceRs(g2);

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("test.dat", 2000, 100, 1, "Wavenumber (cm^-1)", "A_10");
Structure.PlotAbsorptance(g1);

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("Absorptance at 200.dat", 90, 0, .1, "Theta", "T at 200 cm^-1");
Structure.PlotAbsorptance(200, g1);*/

Structure.SetIncidentAngle(30.*pi/180.);

Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("ReflectanceRsSmoothedAngle30.dat", 2000, 100, 1,
    "Wavenumber (cm^-1)", "Rs");
Structure.PlotReflectanceRs(g2);

Structure.IsBulkTreatedCoherent = true;
Gnuplot g3 = Gnuplot("lines");
g3.InitializePlot("ReflectanceRsUnSmoothed.dat", 2000, 100, 1, "Wavenumber (cm^-1)", "Rs");
Structure.PlotReflectanceRs(g3);

Gnuplot g4 = Gnuplot("lines");
g4.InitializePlot("ReflectanceRpUnSmoothed.dat", 2000, 100, 1, "Wavenumber (cm^-1)", "Rp");
Structure.PlotReflectanceRp(g4);

OC.SetNonDispersiveConstantsForLayer(Structure.StackOfLayers.at(0), 3, 0);
OC.ShowOpticalConstants(cout, Structure.StackOfLayers.at(0));

Vacuum.SetPermittivity(1.0);
Vacuum.SetPermeability(1.0);

g1.InitializePlot("RsInt.dat", 90, 0, .1, "Theta", "Transmission, Rs");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "Rs", 200, g1);

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("RpInt.dat", 90, 0, .1, "Theta", "Transmission, Rp");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "Rp", 200, g1);

```

```

Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("TsInt.dat", 90, 0, .1, "Theta", "Transmission, Ts");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "Ts", 200, g2);

Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("TpInt.dat", 90, 0, .1, "Theta", "Transmission, Tp");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "Tp", 200, g2);

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("Rs" + FName + ".dat", 2000, 100, 1, "Theta", "Reflectance, Rs");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Rs", g1, pi*Theta/180.);

g1.InitializePlot("Rp" + FName + ".dat", 2000, 100, 1, "Theta", "Transmission, Rp");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Rp", g1, pi*Theta/180.);

Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("Ts" + FName + ".dat", 2000, 100, 1, "Theta", "Transmission, Ts");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Ts", g2, pi*Theta/180.);

g2.InitializePlot("Tp" + FName + ".dat", 2000, 100, 1, "Theta", "Transmission, Tp");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Tp", g2, pi*Theta/180.);

g1.reset_plot();
g2.reset_plot();

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("rsvl.dat", 90, 0, .1, "Theta", "Transmission, rs");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "rs", 200, g1);

Gnuplot g1 = Gnuplot("lines");
g1.InitializePlot("rpvl.dat", 90, 0, .1, "Theta", "Transmission, rp");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "rp", 200, g1);

Gnuplot g2 = Gnuplot("dash");
g2.InitializePlot("tsvl.dat", 90, 0, .1, "Theta", "Transmission, ts");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "ts", 200, g2);

Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("tpvl.dat", 90, 0, .1, "Theta", "Transmission, tp");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "tp", 200, g2);

g1.InitializePlot("TransInt_s_LtoV.dat", 90, 0, .1, "Wavenumber (cm^-1)",
    "FresnelAmplitude, Ts");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "Ts", 200, g1);
g1.InitializePlot("TransInt_s_VtoL.dat", 90, 0, .1, "Wavenumber (cm^-1)",
    "FresnelAmplitude, Ts");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Ts", 200, g1);

```

```
Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot ("TransInt_p_LtoV.dat", 90, 0, .1, "Wavenumber (cm^-1)",
    "FresnelAmplitude, Tp");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "Tp", 200, g2);
g2.InitializePlot ("TransInt_p_VtoL.dat", 90, 0, .1, "Wavenumber (cm^-1)",
    "FresnelAmplitude, Tp");
Vacuum.PlotFresnelCoefficient (Structure.StackOfLayers.at(0), "Tp", 200, g2);
```

```
Gnuplot g1 = Gnuplot("lines");
Gnuplot g2 = Gnuplot("dash");
```

```
g1.InitializePlot("RefAmp_sWave2.dat", 90, 0, .1, "Wavenumner (cm^-1)",
    "FresnelAmplitude, rs");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "rs", 200, g1);
```

```
g2.InitializePlot("RefAmp_pWave2.dat", 90, 0, .1, "Wavenumner (cm^-1)",
    "FresnelAmplitude, rp");
Structure.StackOfLayers.at(0).PlotFresnelCoefficient(Vacuum, "rp", 200, g2);
```

```
Gnuplot g2 = Gnuplot("lines");
g2.InitializePlot("TransInt_sWave2.dat", 90, 0, .1, "Wavenumner (cm^-1)",
    "FresnelAmplitude, Ts");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Ts", 200, g2);
```

```
g2.InitializePlot("TransInt_pWave2.dat", 90, 0, .1, "Wavenumner (cm^-1)",
    "FresnelAmplitude, Tp");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Tp", 200, g2);
```

```
Gnuplot g3 = Gnuplot("lines");
g3.InitializePlot("Layer1 RefIndex.fat", 2000, 50, 1, "Wavenumner (cm^-1)",
    "Refractive Index");
Structure.StackOfLayers.at(0).PlotRefractiveIndex(g3);
```

```
g2.InitializePlot("RefAmp_p.dat");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "rp", g2);
```

```
g1.reset_plot();
g1.InitializePlot("RefIntensity_p.dat");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Rp", g1);
g1.InitializePlot("TransIntensity_p.dat");
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Tp", g1);
```

```
g1.reset_plot();
g1.InitializePlot("AngleRefAmp_s.dat", 90, 0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "rs", 2000, g1);
```



```

g1.InitializePlot("AngleRefAmp_p.dat", 90, 0.0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "rp", 2000, g1);

g2.reset_plot();
g2.InitializePlot("AngleTransAmp_s.dat", 90, 0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "ts", 2000, g2);
g2.InitializePlot("AngleTransAmp_p.dat", 90, 0.0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "tp", 2000, g2);

Gnuplot g3=Gnuplot("lines");
g3.InitializePlot("AngleTransInt_s.dat", 90, 0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Ts", 2000, g3);
g3.InitializePlot("AngleTransInt_p.dat", 90, 0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Tp", 2000, g3);

Gnuplot g4 = Gnuplot("lines");

g4.InitializePlot("AngleRefInt_s.dat", 90, 0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Rs", 2000, g4);
g4.InitializePlot("AngleRefInt_p.dat", 90, 0, 0.01);
Vacuum.PlotFresnelCoefficient(Structure.StackOfLayers.at(0), "Rp", 2000, g4);

OC.SetNonDispersiveConstantsForStack(Structure);
cout<<"\nStructure.EmergingMedium()\t"<<Structure.GetEpitaxialThickness()<<endl;
Structure.Etch();

return 0;
}

```

A.2 Optical Constants Library

In one of the three methods given for input of optical and electrical constants of dielectric layers, a mathematical expression parsing technique is used to parse constants and operators in mathematical equations saved in a material data file. In other words, the program reads formulas to calculate optical constants. A collection of such files is saved in a directory (for example “c:/Optical Constants Library/”). The following is an example of layout for a given material.

```
% Optical Constants for GaAs Epitaxial films
% Updated on January 09, 2006
```

High Frequency Dielectric Constant(ϵ_s/ϵ_0)	= 10.6039
Static Dielectric Constant($\epsilon_\infty/\epsilon_0$)	= 12.4062
Electron Effective Mass(m_e/m_0)	= 0.0667
Hole Effective Mass(m_h/m_0)	= 0.5

% For Photon-Plasma interaction	
Resonance Frequency/cm ⁻¹ (ω_{plasmon} /cm ⁻¹)	= 410.
Strength (D_{plasmon})	= 100.

% For Photon-Phonon Interaction	
Resonance Frequency (ω_{TO} /cm ⁻¹)	= 554.5
Strength (S_{TO})	= 15.
Damping (γ_{TO} /cm ⁻¹)	= 10.
Resonance Frequency (ω_{LO} /cm ⁻¹)	= 550.
Strength (S_{LO})	= 15.
Damping (γ_{LO} /cm ⁻¹)	= 10.

```

Resonance Frequency (W_TO/cm^-1)      = 554.5
Strength (S_TO)                        = 15.
Damping (D_TO/cm^-1)                  = 10.
Resonance Frequency (W_LO/cm^-1)      = 550. * (1-4.0e-5*T) + 1.83*x
                                         + 17.12*pow(x,2) - 5.11*pow(x,3)
Strength (S_LO)                        = 15.
Damping (D_LO/cm^-1)                  = 10.

```

```
% For Photon-Phonon Interaction
```

```

% Resonance Frequency (W_TO/cm^-1)    = 554.5
% Strength (S_TO)                     = 15.
%Damping (D_TO/cm^-1)                 = 10.
%Resonance Frequency (W_LO/cm^-1)    = 550.
%Strength (S_LO)                      = 15.
%Damping (D_LO/cm^-1)                 = 10.

```

Note: The program can read up to 10 TO-LO pairs starting from the top, and this number can be increased by changing the static variable of class “DielectricLayer::PhononSize”. By placing a ‘%’ symbol at the start of the line (as shown in the above example), any given phonon(s) can be masked from being read by the program. The labels ‘x’ and ‘T’ in the equations are alloy composition of the material and the dielectric stack temperature, respectively.

A.3 Digital Filtering of Interference Fringes

To remove fringe effects (only if necessary) caused by thick substrates, the “Savitzky-Golay” digital filtering is used. The best order of polynomial for the fits was found to be 4. User call of the “Digital Filter” any number of times to obtain the preferred degree of smoothing is permitted. However, overusing the smoothing function can cause the peak

around the reststrahlen region to reduce considerably. The effect of digital filtering in removing the fringes from a model response spectra is shown in Fig. A.1. In this case, the substrate thickness is $400\text{ }\mu\text{m}$. The ‘ x -axis’ was selected to be in wavenumber to show the regularity of the interference pattern. A similar set of curves is shown in Fig. A.2. The smoothing with different combinations of data points and the number of passes show that by increasing the number, the number of passes needed for reasonable smoothing can be reduced.

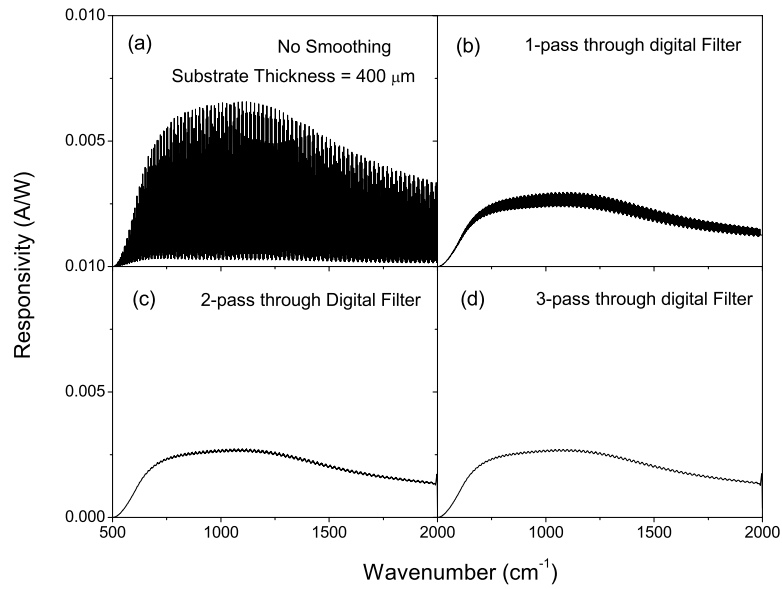


Figure A.1: Effect of Savitzky-Golay digital filtering on the responsivity of a detector. The total number of data points used was 21. (a) The contribution to responsivity from top contact without any smoothing and smoothing with (b) a single pass, (c) double passes, and (d) triple passes through the filter.

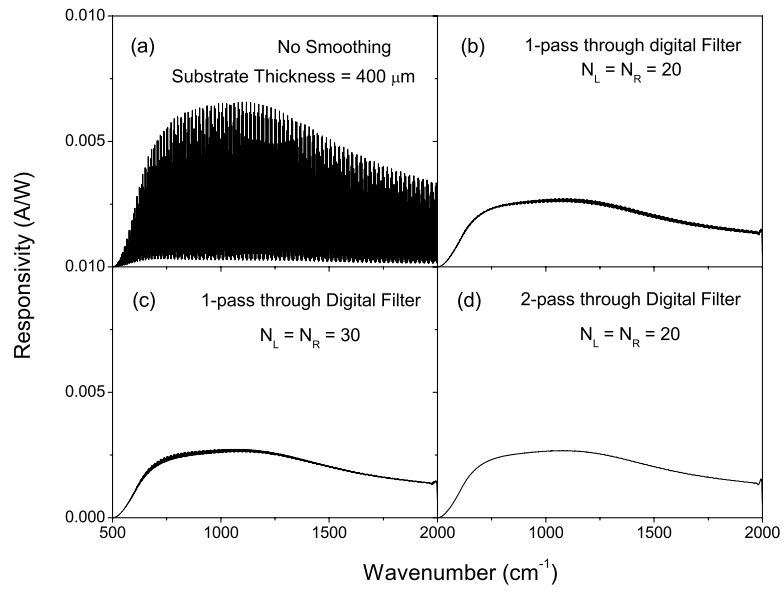


Figure A.2: A combination of number of adjacent points used in the moving window for smoothing, and the number of passes through the filter. N_L and N_R indicate the number of left and right adjacent data points. (a) The contribution to responsivity from top contact without any smoothing and smoothing with (b) a single pass with $N_L = N_R = 20$, (c) a single pass with $N_L = N_R = 30$, and (d) double passes with $N_L = N_R = 20$ through the filter.

Appendix B

Programs Used for Switching in Heterostructures

The following program is written in C++ (version 6.0). The program is a collection of classes that allow the evaluation of current through a heterostructure for a given bias and temperature. The current for a given bias is calculated through an iteration process that terminates when the current or the electron temperature value converges to a desired precision. The coding for the iterative evaluation of these electron temperatures and currents for a given lattice temperature is given in the main body of the program. The main classes used were as follows.

- Carrier
- Layer
- Barrier
- Integral
- Device

The class “Layer” is not similar to the class “Dielectric Layer”, which is defined in **Appendix A**. The “Barrier” class is derived from the “Layer” class. Therefore, in addition to the functions defined in “Barrier.h”, all the functions in the “Layer” class will be accessible by “Barrier” objects. The adaptive quadrature algorithm used in the class “Integral” was adapted from the book “Numerical Recipes in C++: The Art of Scientific Computing” by William H.Press. The use of a function pointer in the class “Integral” allows easy evaluation of integrals of functions within a given lower and upper bounds, and a resolution. The function only contains the expression that needs to be evaluated. Several equation solver routines have been added in the main program. The algorithms for these were adapted from the above reference.

//Copyright - M. B. Rinzan, Optoelectronics Lab – Georgia State University

//Class declaration for Carriers

```
#ifndef CARRIER_H
#define CARRIER_H
```

```
#include "Layer.h"
#include "Constants.h"
#include "math.h"
```

```
struct vector{
    long double x;
    long double y;
    long double z;
    long double mag;
};
```

```
class Carrier
{
```

```
    private:
```

```
        static long double charge;
        static long double mass;
        char type;
```

```
    public:
```

```
        Carrier(char t='?', long double d=0., long double T = 0.);
        Carrier(Layer layer);
        ~Carrier(){};
        void ListContent(void);
        long double density;
        long double effective_mass;
        long double mobility;
        long double scattering_time;
        long double temperature;
        vector wave;
        vector drift;
```

```
        void set_type(char t){type=t;};
        long double SetMobility(Layer &layer);
        long double set_effective_mass(Layer &L);
        void set_scattering_time(Layer L);
```

```
        inline void SetDrift(long double Dx, long double Dy, long double Dz)
        {
            drift.x=Dx; drift.y=Dy; drift.z=Dz;
            drift.mag = pow(Dx*Dx + Dy*Dy + Dz*Dz,0.5);
        };
```



```
inline void SetWave (long double Wx, long double Wy, long double Wz)
{
    wave.x=Wx; wave.y=Wy; wave.z=Wz;
    wave.mag = pow(Wx*Wx + Wy*Wy + Wz*Wz,0.5);
};

long double _mass(void){return mass;};
long double _charge(void){return charge;};
char _type(void){return type;};
};

#endif
```

```
//Implementation of Class Carrier
```

```
#include "stdafx.h"
#include "Carrier.h"
```

```
double Carrier::Charge    = 1.602e-19;      //C
double Carrier::FreeMass  = 9.10938188e-31;  //kg
```

```
Carrier::
Carrier(char _Type, double _Density, double Effective_Mass, double _Mobility,
        double Scattering_Time, double _Temperature)
        :Type(_Type),
          Density(_Density),
          EffectiveMass(Effective_Mass),
          Mobility(_Mobility),
          ScatteringTime(Scattering_Time),
          Temperature(_Temperature),
          DriftWaveVector(),
          RandomWaveVector() {}
```

```
void Carrier::Erase(void)
{
    Type = '?';
    Density = 0.;
    EffectiveMass = 0.;
    Mobility = 0.;
    ScatteringTime = 0.;
    Temperature = 0.;
    DriftWaveVector.Erase();
    RandomWaveVector.Erase();
}
```

```
void Carrier::ListContent(void)
{
    cout << *this << endl
        << "Effective Mass:\t" << EffectiveMass << endl
        << "Temperature:\t" << Temperature << endl
        << "Mobility:\t" << Mobility << endl
        << "Scattering Time:\t" << ScatteringTime << endl
        << "Carrier Drift(cm^-1):\n" << DriftWaveVector << endl;
}
```

```
void Carrier::SetEffectiveMass(const double &Effective_Mass)
{
    EffectiveMass = Effective_Mass;
}
```

```
Carrier::Carrier(char t, long double d, long double T)
```

```
{
    type = t;
    density = d;
    temperature = T;
    wave.x = 0.;
    wave.y = 0.;
    wave.z = 0.;
    wave.mag = 0.;

    drift.x = 0.;
    drift.y = 0.;
    drift.z = 0.;
    drift.mag = 0.;
}
```

```
Carrier::Carrier(Layer layer)
```

```
{
    type = layer._dopant();
    density = layer._doping_density();
    temperature = layer.temperature;
    SetMobility(layer);
    set_effective_mass(layer);

    wave.x = 0.;
    wave.y = 0.;
    wave.z = 0.;
    wave.mag = 0.;

    drift.x = 0.;
    drift.y = 0.;
    drift.z = 0.;
    drift.mag = 0.;
}
```

```
void Carrier::ListContent(void)
```

```
{
    cout<<"\n\tDensity:\t"<<density;
    cout<<"\n\tType:\t"<<type;
    cout<<"\n\tEffective Mass:\t"<<effective_mass;
    cout<<"\n\tTemperature:\t"<<temperature;
    cout<<"\n\tMobility:\t"<<mobility;
    cout<<"\n\tcarrier is set to:\t(" <<wave.x<<"," <<wave.y<<"," <<wave.z<<) m^-1";
    cout<<"\n\tDrift due to bias:\t(" <<drift.x<<"," <<drift.y<<"," <<drift.z<<) m^-1";
}
```

```

//Definition of Barrier class derived from the Layer class
#ifndef BARRIER_H
#define BARRIER_H

#include "Layer.h"
#include "Carrier.h"

class Barrier:public Layer{
private:
    long double BandOffset, bias, EFermi;
    long double a, z0, zS, zPrime, k1, k3;

public:
    Barrier(string t="NONE", string b="NONE", long double w=0., char d='?',
            long double dt=0., string a="NONE", long double f=0.)
        :Layer(t,b,w,d,dt,a,f){ };

    long double      SetBandOffset (Layer &layer);
    void              SetBias (long double V);
    long double      GetBandOffset (void);
    long double      GetBias (void);

    long double      GetTunnelingCurrent
        (string shape, string method, Layer &Well, long double fld,
         long double kD);

    long double      GetTunnelingCurrent
        (class Carrier carrier, string shape, string method, long double barrier,
         long double mW, long double mB, long double V, long double W);

    long double      GetThermalCurrent
        (Layer &well, long double ElectricalArea, long double bBias);

    void              PlotThermal Current
        (Layer &Injector, long double ElectricalArea, string FileName,
         long double LowerLimit, long double UpperLimit, long double Step);

    long double      GetDistribution
        (long double CarrierDensity, long double WellMass,
         long double Temperature, long double zOfWave, long double OfDrift);

    long double      GetTransmission
        (string shape, string method, long double D, long double Wz,
         long double mW, long double mB, long double field);

};

#endif

```

```

//Implementation of the Barrier class

#include "Barrier.h"
#include "Carrier.h"
#include "Constants.h"
#include "airy.h"
#include "layer.h"
#include "integral.h"

namespace TUNNEL {

    long double n, T, mW, mB, Delta, kD, FB;
    string shape, method;
    Barrier bb;

    long double _cdecl GetTunnelingCurrent(long double kz) {
        return 2*e*bb.GetDistribution(n, mW, T, kz, kD)*
            bb.GetTransmission(shape, method, Delta, kz, mW, mB, fabs(FB));
    };

}

long double Barrier::GetThermalCurrent (Layer &well, long double A, long double V)
{
    long double TotalOffset, Elec_Field, M, mW;
    Carrier electron;
    V = V*thickness/(thickness+well._thickness());
    Elec_Field = V/thickness;
    SetBandGap(); //Sets Band Gap of Barrier
    well.SetBandGap(); //Sets Band Gap of Well
    TotalOffset = (BandGap-well._BandGap())*e; //Total Band Gap Offset
    mW = electron.set_effective_mass(well); //Assigns carrier mass in Well to mW

    electron.SetMobility(well);
    M = electron.mobility;
    long double vSat = 1.e5;
    long double alpha = 0.;

    return e*A*M*Elec_Field*2*pow(2*p*mW*k*well.temperature/pow(h,2),1.5)
        *exp(-(TotalOffset-e*V)/(k*well.temperature))
        /pow(1+pow(M*Elec_Field/vSat,2),0.5);
}

```

```
void Barrier::PlotThermalCurrent(Layer& Injector, long double A, string FileName, long double Lower,
long double Upper, long double Step)
```

```
{
    ofstream Out;
    long double Bias;
    Out.open(FileName.c_str(), ios::out);
    if (Lower > Upper) {
        Bias    = Lower;
        Lower   = Upper;
        Upper   = Bias;
    }

    Bias = Lower;
    while (Bias <= Upper) {
        Out<<Bias<<"\t"<<GetThermalCurrent(Injector, A, Bias)<<endl;
        Bias += Step;
    }
}
```

```
long double Barrier::SetBandOffset(Layer &well)
```

```
{
    long double TotalOffset;

    SetBandGap();
    well.SetBandGap();
    TotalOffset = BandGap-well._BandGap();

    //Evaluating the band offset
    if (well._base()=="GaAs" && base=="GaAs"){
        if (well._dopant()=='p')
            BandOffset = 0.4 * TotalOffset; else
            BandOffset = 0.6 * TotalOffset;}
    else if (well._base()=="xxx" && base=="xxx"){
        if (well._dopant()=='x')
            BandOffset = 0. * TotalOffset; else
            BandOffset = 0. * TotalOffset;}

    return BandOffset*e;
}
```

```
void Barrier::SetBias(long double V)
```

```
{
    bias = V;
}
```

```
long double Barrier::GetBias(void)
```

```
{
    return bias;
}
```

```

long double Barrier::GetBandOffset(void)
{
    return BandOffset;
}

long double Barrier::GetTunnelingCurrent(string shape, string method, Layer &well, long double Barfield,
long double Emfield)
{
    Carrier carrier(well._dopant(), well._doping_density());
    long double zInitial, zFinal, zStep, DriftVector;
    long double mB, mW, temp1, temp2, sum;
    fstream file;

    zInitial = 1.e5;
    zFinal = 1.e10;
    zStep = 1.e5;

    carrier.temperature = well.temperature;
    SetBandOffset(well);
    mW = carrier.set_effective_mass(well); //Assigns carrier mass in Well to mW
    mB = carrier.set_effective_mass(*this); //Assigns carrier mass in Barrier to mB
    carrier.SetMobility(well); //Sets carrier mobility in Well
    DriftVector = (carrier.mobility*carrier.effective_mass/hBar)*Emfield;
    carrier.SetWave(0, 0, zInitial);
    carrier.SetDrift(0, 0, DriftVector);

    //Set TransmissionTest <> 0 to check transmission
    int TransmissionTest = 1;
    if (TransmissionTest){
        carrier.ListContent();
        file.open("TransmissionTest.dat",ios::out);
        do{
            file<<carrier.wave.z<<"t"<<
                GetTransmission(method, BandOffset, carrier.wave.z, mB,field)<<endl;
            file<<(pow(hBar*carrier.wave.z,2)/(2.*mW))/e<<"t"<<
                GetTransmission(shape, method, BandOffset, carrier.wave.z, mW, mB,
                Barfield)
                <<endl;
            carrier.wave.z+=zStep;
        }while (carrier.wave.z<=zFinal);
        file.close();
        carrier.SetWave(0, 0, zInitial);
        exit(0);
    }
    //Transmission test ends

    //Evaluating tunneling
    sum = 0.;
    temp1 = GetDistribution(carrier.density, mW, carrier.temperature, carrier.wave.z, carrier.drift.z);
    temp1 *= GetTransmission(shape, method, BandOffset, carrier.wave.z, mW, mB, Barfield);

```

```

do{
    //Set CurrDenSpecTest <> 0 to check Current density spectra
    int CurrDenSpecTest = 1;
    if (CurrDenSpecTest){
        carrier.ListContent();
        file.open("CurrDenSpecTest.dat",ios::out);
        file<<temperature<<endl;
        do{
            carrier.wave.z += zStep;
            temp2 = GetDistribution (carrier.density, mW,
                                   carrier.temperature, carrier.wave.z, carrier.drift.z);

            temp2 *= GetTransmission(shape, method, BandOffset,
                                    carrier.wave.z, mW, mB, Barfield);

            file<<(pow(hBar*carrier.wave.z,2)/(2.*mW))/e<<"\t"<<temp2<<endl;
            carrier.wave.z+=zStep;

        } while (carrier.wave.z<=zFinal);

        file.close();
        carrier.SetWave(0, 0, zInitial);
        exit(0);
    }
    //CurrDenSpecTest

    carrier.wave.z += zStep;
    temp2 = GetDistribution
        (carrier.density, mW, carrier.temperature, carrier.wave.z, carrier.drift.z);
    temp2 *= GetTransmission
        (shape, method, BandOffset, carrier.wave.z, mW, mB, Barfield);
    sum += 0.5*(temp1 + temp2)*zStep;
    temp2 = temp1;

}while (carrier.wave.z<=zFinal);

return sum;
}

long double Barrier::GetTunnelingCurrent
    (Carrier carrier, string shape, string method, long double barrier,
     long double mW, long double mB, long double Voltage, long double Emitter_Thickness)
{
    TUNNEL::bb=*this;
    TUNNEL::Delta = BandOffset;
    TUNNEL::method = method;
    TUNNEL::shape = shape;
    TUNNEL::mW = mW;
    TUNNEL::mB = mB;
    TUNNEL::T = carrier.temperature;

```



```

TUNNEL::n = carrier.density;
TUNNEL::kD = carrier.drift.z;
TUNNEL::FB = (Voltage-Emitter_Thickness*hBar*TUNNEL::kD/
              (TUNNEL::mW*carrier.mobility))/_thickness();

long double zInitial      = 1.;
long double zFinal        = 1.e20;
long double tolerance     = 0.001;

Integral I_tunnel(zInitial,zFinal, TUNNEL::GetTunnelingCurrent);
//I_tunnel.PlotIntegrand("test.dat", carrier.drift.z, 100000);
/I_tunnel.PlotIntegrand("test1.dat", carrier.drift.z, "log", 100000);

return I_tunnel.Trapezoidal(200000);
}

long double Barrier::
GetDistribution (long double n, long double m, long double T, long double kz, long double kd)
{
    return 0.5*p*k*T*n*hBar*hBar*pow(2*p*m*k*T, -1.5)*kz*exp(-hBar*hBar*
        (kz-kd)*(kz-kd)/(2*m*k*T));
}

long double Barrier::
GetTransmission (string shape, string method, long double D, long double Wz,
                 long double mW, long double mB, long double F)
{
    long double Ez, KF, Eta_0, a, tmp, temp1, temp2, temp3, temp4, mrat;

    Ez = pow(hBar*Wz,2)/(2.*mW);
    KF = pow(2.*mW*e*F/pow(hBar,2), 1./2.);
    Eta_0 = KF*(D-Ez)/(e*F);
    if (method=="WKB"){
        if (Eta_0>1.)
            //For the time being I have removed reflections for Energy above the Barrier
            return (4./D)*pow(Ez*(D-Ez), 0.5)*exp(-4.*pow(Eta_0,1.5))/3.;
        else
            4.*pow((Ez-D)/Ez, 0.5)/pow(1. + pow((Ez-D)/Ez, 0.5), 2.);}

    else if (method=="Gundlach"){
        if (shape=="Triangular"){
            temp1 = pow(get_Bi(Eta_0)-(KF/Wz)*get_Ai_prime(Eta_0), 2);
            temp2 = pow(get_Ai(Eta_0)+(KF/Wz)*get_Bi_prime(Eta_0),2);
            return (4.*KF/(p*Wz))*(1./(temp1+ temp2));
        }
    }
}

```

```

else if (shape=="Trapezoidal"){
    mrat = mB/mW;
    int correction=0;
    if (!correction){
        mrat = 1;
        mW=mB;};
    a = 2.*sqrt(2.*mB)/hBar;
    tmp = pow(a/(2.*e*F),2./3.);
    z0 = tmp*(BandOffset-Ez);
    zS = tmp*(BandOffset-Ez-e*F*thickness);
    zPrime = -pow(a*a/4.*e*F,1./3.);
    a = 2. *sqrt(2.*mW)/hBar;
    k1 = a/2.*sqrt(Ez);
    k3 = a/2.*sqrt(Ez+e*F*thickness);

    if ((z0 < 50) || (zS < 50)){
        temp1 = get_Ai_prime(z0)*get_Bi_prime(zS)-
            get_Ai_prime(zS)*get_Bi_prime(z0);
        temp2 = get_Ai(z0)*get_Bi(zS)-get_Ai(zS)*get_Bi(z0);
        temp3 = get_Ai(zS)*get_Bi_prime(z0)-get_Ai_prime(z0)*get_Bi(zS);

        temp4 = get_Ai(z0)*get_Bi_prime(zS)-get_Ai_prime(zS)*get_Bi(z0);
        tmp = zPrime*temp1/(k1*mrat)+mrat*k3*temp2/zPrime;
        tmp = tmp*tmp;
        temp1 = k3*temp3/k1 + temp4;
        temp1 = temp1*temp1;
        temp2 = 1./(tmp+temp1);
        temp3 = temp2*k3*4./(k1*p*p);
        return temp3;
    }
    temp1 = -2.*a*thickness/3.;
    temp2 = pow(BandOffset-Ez, 1.5);
    temp3 = pow(BandOffset-Ez-e*F*thickness,1.5);
    tmp = exp(temp1*(temp2-temp3)/(e*F*thickness));
    temp1 = sqrt((BandOffset-Ez)*fabs(e*F*thickness-(BandOffset-Ez)))*
        Ez*k3/(k1*pow(BandOffset,2));
    return 16.*tmp*temp1;
}
else
    return -1;
}
else
    return -1.;
}

```

```

//Definition of class Integral

#ifndef Integral_H
#define Integral_H

#include "date.h"
#include <iostream>
#include <fstream>
#include <complex>
#include <string>
#include <conio.h>

using namespace std;

typedef long double (*ptr_func) (long double);

class Integral{
private:
    long double lower;
    long double upper;
    ptr_func integrand;
public:
    Integral (long double a, long double b, ptr_func f) {lower = a; upper =b; integrand = f;};

    long double lower_bound (void) const {return lower;};

    long double upper_bound (void) const {return upper;};

    void set_bound (long double LB, long double UB) {lower = LB; upper = UB;};

    \
    void PlotIntegrand (string FileNameToSavePlot, long double DriftVector, int RefineSize);

    void PlotIntegrand (string FileNameToSavePlot, long double DriftVector,
                        string step_type="log or linear", int DataSize=10000);

    long double AdaptiveQuadrature
        (string SegmentType = " `trapezoidal' or `simpson' ", long double Precision = 0.001);

    long double ModifiedAdaptiveQuadrature
        (string SegmentType = " `trapezoidal' or `simpson' ", long double Precision= 0.001,
         int MaxStepSize = 1000);

    long double Trapezoidal(int);
    long double Simpson (int, long double);
};

#endif

```

//Implementation of the Integral class

```
#include "integral.h"
#include "time.h"
#include "date.h"
extern time_of_day now;
extern date today;

void Integral::PlotIntegrand(string FileName, long double kD, int size)
{
    long double x, Xmax, Ymax, xL, xR;
    long double step;
    int count = 0;
    fstream file;

    today.set_date_format(4);
    file.open(FileName.c_str(), ios::out);
    file<<"\tFile generated on "<<today.get_date_string()<<" at "<<now.get_time_string()<<endl;
    file<<"\tIntegrand: Tunneling current through the barrier"<<endl;
    file<<"\tCarrier drift due to field and barrier lowering: kD = "<<kD<<" m^-1"<<endl;

    x = lower;
    step = lower;
    Ymax = -1.;

    do {
        if (integrand(x)>Ymax){
            Ymax = integrand(x);
            Xmax = x;
        }
        x+=step;
        if (count==9) {
            step=step*10;
            count=0;
        }
        count++;
    } while(x<upper);

    xL = 0.1*Xmax; xR = 10*Xmax;

    file<<"\tRefined region : "<<xL<<" - "<<xR<<endl;
    file<<"\tNumber of fine steps = "<<size<<endl;

    x = lower;
    step = lower;
    count = 0;
    do {
        file<<x<<"\t"<<integrand(x)<<endl;
        x+=step;
        if (count==9) {
```

```

        step=step*10;
        count=0;
    }
    count++;
}while(x<xL);

x = xL;
step = (xR-xL)/size;
do {
    file<<x<<"\t"<<integrand(x)<<endl;
    x+=step;
}while (x<xR);

x = xR;
step = x;
count = 0;
do {
    file<<x<<"\t"<<integrand(x)<<endl;
    x+=step;
    if (count==9) {
        step=step*10;
        count=0;
    }
    count++;
}while (x<upper);

cout<<"\nPlot completed"<<endl;
}

void Integral::PlotIntegrand(string FileName, long double kD, string step_type, int size)
{
    long double x, step;
    int count =0;
    fstream file;

    today.set_date_format(4);
    file.open(FileName.c_str(), ios::out);
    file<<"\tFile generated on "<<today.get_date_string()<<" at "<<now.get_time_string()<<endl;
    file<<"\tIntegrand: Tunneling current through the barrier"<<endl;
    file<<"\tCarrier drift due to field and barrier lowering: kD = "<<kD<<" m^-1"<<endl;

    //Plotting the Integrand;

    if (step_type=="log") {
        x = lower;
        step = lower;
        count =0;

```

```

        do {
            file<<x<<"\t"<<integrand(x)<<endl;
            x+=step;
            if (count==9) {
                step=step*10;
                count=0;
            }
            count++;
        }while(x<=upper);
    }
    else if (step_type == "linear") {
        x = lower;
        step = fabs(upper-lower)/long double(size);
        do {
            file<<x<<"\t"<<integrand(x)<<endl;
            x+=step;
        }while(x<=upper);
    }
    else {
        cout<<"Step type \"<<step_type<<\" is incorrect or not implemented\n";
        file<<"Step type \"<<step_type<<\" is incorrect or not implemented\n";
        exit(0);
    }

    cout<<"\nPlot completed"<<endl;
}

```

```

long double Integral::ModifiedAdaptiveQuadrature(string SegmentType, long double Tolerance, int Level)
{
    long double x, XYmax, step, xL, xR, sum = 0.;
    int count =0;

    x = 0.;
    step = lower;
    XYmax = 1e-6;

    do {
        x+=step;
        if (count==9) {
            step=step*10;
            count=0;
        }
        count++;
    }while(x*integrand(x)<XYmax);

    cout << (xL = x) <<endl;
}

```

```

do {
    x+=step;
    if (count==9) {
        step=step*10;
        count=0;
    }
    count++;
}while (x*integrand(x)>XYmax);

cout <<(xR = x) <<endl;;

set_bound(xL, xR);

const max_level = 5000;
long double s[max_level], a[max_level], h[max_level], FA[max_level], FB[max_level],
            FC[max_level], L[max_level], TOL[max_level], v[8], FD, FE, S1, S2;
long double App = 0;
int i = 0;

TOL[i] = 10.*Tolerance;
a[i] = lower;
h[i] = (upper-lower)/2.;
FA[i] = integrand(lower);
FB[i] = integrand(upper);
FC[i] = integrand(lower+h[i]);

s[i] = (h[i]/3.)*(FA[i]+4*FC[i]+FB[i]);    //Approximation for the entire interval
L[i] = 1;

while (i>-1) {
    FD = integrand(a[i] + h[i]/2.);
    FE = integrand(a[i] + 3.*h[i]/2.);
    S1 = (h[i]/6.)*(FA[i]+4*FD+FC[i]);
    //Approximation for halves of sub-intervals
    S2 = (h[i]/6.)*(FC[i]+4*FE+FB[i]);
    v[0] = a[i];
    v[1] = FA[i];
    v[2] = FC[i];
    v[3] = FB[i];
    v[4] = h[i];
    v[5] = TOL[i];
    v[6] = s[i];
    v[7] = L[i];
    i -=1;
}

```

```

        if(fabs(S1+S2-v[6])<v[5])
            App += (S1+S2);
        else {
            if(v[7]>=Level) {
                cout<<"\n\tLevel exceeded "<<Level;
                return App;
            }
            if(v[7]>=max_level) {
                cout<<"\n\tLevel exceeded the storage allocated for Simpson "
                    <<max_level;
                return App;
            }
            else {
                //Data for left half sub-interval
                i +=1;
                a[i] = v[0]+v[4];
                FA[i] = v[2];
                FC[i] = FE;
                FB[i] = v[3];
                h[i] = v[4]/2.;
                TOL[i] = v[5]/2.;
                s[i] = S2;
                L[i] = v[7]+1;

                //Data for right half sub-interval
                i +=1;
                a[i] = v[0];
                FA[i] = v[1];
                FC[i] = FD;
                FB[i] = v[2];
                h[i] = h[i-1];
                TOL[i] = TOL[i-1];
                s[i] = S1;
                L[i] = L[i-1];
            }
        }
    }
    return App;
}

```

```

long double Integral::Trapezoidal(int size)
{
    long double x1, x2, x, Xmax, Ymax, step, xL, xR, sum = 0.;
    int count =0;
    x = lower;
    step = lower;
    Ymax = -1.;

```



```

do {
    if (integrand(x)>Ymax){
        Ymax = integrand(x);
        Xmax = x;
    }
    x+=step;
    if (count==9) {
        step=step*10;
        count=0;
    }
    count++;
}while (x<upper);

//Refining the peak region of the function
xL = 0.1*Xmax;
xR = 10*Xmax;

//Calculating the integral;
x1 = lower;
step =lower;
sum = 0.;

do {
    x2 = x1+step;
    sum+=(0.5*(integrand(x1)+integrand(x2))*step);
    x1 = x2;
    if (count==9) {
        step=step*10;
        count=0;
    }
    count++;
} while(x2<xL);

x1 = xL;
step = (xR-xL)/long double(size);

do {
    x2 = x1+step;
    sum+=(0.5*(integrand(x1)+integrand(x2))*step);
    x1 = x2;
} while(x2<xR);

x1 = xR;
step = x1;

```

```

do {
    x2 = x1+step;
    sum+=(0.5*(integrand(x1)+integrand(x2))*step);
    x1 = x2;
    if (count==9) {
        step=step*10;
        count=0;
    }
    count++;
} while(x2<xR);

return sum;
}

```

```

long double Integral::Simpson(int Level, long double Tolerance)
{
    long double x, Xmax, Ymax, step, xL, xR, sum = 0.;
    int count =0;

    x = lower;
    step = lower;
    Ymax = -1.;

    do {
        if (integrand(x)>Ymax){
            Ymax = integrand(x);
            Xmax = x;
        }
        x+=step;
        if (count==9) {
            step=step*10;
            count=0;
        }
        count++;
    }while(x<upper);

    //Refining the boundary for Simpson
    xL = 0.1*Xmax;
    xR = 5*Xmax;
    set_bound(xL, xR);

    const max_level = 1000;
    long double s[max_level], a[max_level], h[max_level], FA[max_level],
        FB[max_level], FC[max_level];
    long double L[max_level], TOL[max_level], v[8], FD, FE, S1, S2;
    long double App = 0;
    int i = 0;

```

```

TOL[i] = 10.*Tolerance;
a[i] = lower;
h[i] = (upper-lower)/2.;
FA[i] = integrand(lower);
FB[i] = integrand(upper);
FC[i] = integrand(lower+h[i]);
s[i] = (h[i]/3.)*(FA[i]+4*FC[i]+FB[i]);           //Approximation for the entire interval
L[i] = 1;

while (i>-1) {
    FD = integrand(a[i] + h[i]/2.);
    FE = integrand(a[i] + 3.*h[i]/2.);
    S1 = (h[i]/6.)*(FA[i]+4*FD+FC[i]);           //Approximation for halves of sub-intervals
    S2 = (h[i]/6.)*(FC[i]+4*FE+FB[i]);
    v[0] = a[i];
    v[1] = FA[i];
    v[2] = FC[i];
    v[3] = FB[i];
    v[4] = h[i];
    v[5] = TOL[i];
    v[6] = s[i];
    v[7] = L[i];
    i -=1;

    if(fabs(S1+S2-v[6])<v[5])
        App += (S1+S2);
    else {
        if(v[7]>=Level) {
            cout<<"\n\tLevel exceeded "<<Level;
            return App;
        }
        if(v[7]>=max_level) {
            cout <<"\n\tLevel exceeded the storage allocated for Simpson "
                <<max_level;
            return App;
        }
        else {
            //Data for left half sub-interval
            i +=1;
            a[i] = v[0]+v[4];
            FA[i] = v[2];
            FC[i] = FE;
            FB[i] = v[3];
            h[i] = v[4]/2.;
            TOL[i] = v[5]/2.;
            s[i] = S2;
            L[i] = v[7]+1;

            //Data for right half sub-interval

```

```
        i +=1;
        a[i] = v[0];
        FA[i] = v[1];
        FC[i] = FD;
        FB[i] = v[2];
        h[i] = h[i-1];
        TOL[i] = TOL[i-1];
        s[i] = S1;
        L[i] = L[i-1];
    }
}
return App;
}
```

```

//Declaration for the Layer Class
#ifndef LAYER_H
#define LAYER_H

#include <iostream>
#include <stdlib.h>
#include <string>

using namespace std;

//The number of allowed modes for LO and TO phonons
const int No_of_modes=10;

//Ordering the base material for the switch argument
enum base_materials{
    GaAs,
    GaN,
    Sapphire,
    /*additional materials should be added before this line. Remember to modify the relevant
    portions in the header*/
};

struct oscillator{
    long double resonance_frequency;
    long double amplitude;
    long double damping;
};

class Layer{
protected:
    string type;
    string base;
    string alloy;
    long double fraction;
    long double thickness;
    char dopant;
    long double doping_density;
    string defect;
    long double defect_density;
    long double BandGap;
public:
    Layer (string t="NONE", string b="NONE", long double w=0., char d='?',
           long double dty=0., string a="NONE", long double f=0.);
    ~Layer(){}
    void ListLayerContent (void);

    long double static_index;
    long double high_index;
    long double temperature;
    long double Debye_temperature;

```

```

oscillator      plasmon;
oscillator      phonon [No_of_modes];
void            set_type (string t)      {type=t;};
int             material (string material);

void            set_base (string b)      {base=b;};
void            set_alloy (string a)     {alloy=a;};
void            set_fraction (long double f) {fraction=f;};
void            set_thickness (long double w) {thickness=w;};
void            set_dopant (char d)      {dopant=d;};
void            set_doping_density (long double n) {doping_density=n;};
void            set_defect (string def)   {defect=def;};
void            set_defect_density (long double defn)
              {defect_density=defn;};

long            double k_zo
              (Layer barrier, class Carrier &carrier, long double Voltage, long double kd);

string          SetBandGap (void);
string          _type (void)             {return type;};
string          _base (void)             {return base;};
string          _alloy (void)            {return alloy;};

long double     _fraction (void)         {return fraction;};
long double     _thickness (void)        {return thickness;};
char            _dopant (void)           {return dopant;};
long double     _doping_density (void)   {return doping_density;};
string          _defect (void)           {return defect;};
long double     _defect_density (void)   {return defect_density;};
long double     _BandGap (void)          {return BandGap;};
};
#endif

```

```
//Implementation of class "Layer"
```

```
#include "Layer.h"
#include <string.h>
#include <fstream>
#include <conio.h>
#include <math.h>
#include "Constants.h"
#include "Carrier.h"
```

```
Layer::Layer(string t, string b, long double w, char d, long double dty, string a, long double f)
```

```
{
    type=t;
    base=b;
    alloy=a;
    fraction=f;
    thickness=w;
    dopant=d;
    doping_density=dty;
    defect='?';
    defect_density=0.;
    static_index=12.40 - 2.84*f;
    high_index=0.;
    temperature=0.;
    Debye_temperature=0.;

    plasmon.amplitude=0.;
    plasmon.resonance_frequency=0.;
    plasmon.damping=0.;

    for (int i=0;i<No_of_modes;i++){
        phonon[i].amplitude=0.;
        phonon[i].resonance_frequency=0.;
        phonon[i].damping=0.;
    }
}
```

```
void Layer::ListLayerContent(void)
```

```
{
    cout<<"\n\ttype"<<type;
    cout<<"\n\tbase"<<base;
    cout<<"\n\talloy"<<alloy;
    cout<<"\n\tfraction"<<fraction;
    cout<<"\n\tthickness"<<thickness;
    cout<<"\n\tdopant"<<dopant;
    cout<<"\n\tdoping_density"<<doping_density;
}
```

```

int Layer::material(string mat)
{
    if (mat=="GaAs")
        return GaAs;
    else if (mat=="GaN")
        return GaN;
    else if (mat=="Sapphire")
        return Sapphire;
    else{
        cerr<<"\n\t"<<base<<" is not in the material list";
        exit(UNDEFINED_MATERIAL);
    }
}

string Layer::SetBandGap(void)
{
    string Transition;
    switch(material(base)){
        case GaAs:
            if(alloy=="NONE") {
                BandGap = 1.519-5.405e-4*pow(temperature,2)/(temperature+204);
                Transition = "Direct";
            }
            //Al alloy
            else if(alloy=="Al") {
                //G-valley transition
                if(fraction<=0.45){
                    BandGap = 1.519 + 1.247*fraction - 5.41e-4
                        *pow(temperature,2)/(temperature+204);
                    Transition = "Direct";
                }
                //L-valley transition
                else{
                    BandGap = 1.8145 + 0.6725*fraction;
                    BandGap -= 6.05e-4*pow(temperature,2)*
                        ((1.-fraction)/(temperature+204.))+1.304*
                        fraction/(temperature+208.);
                    Transition = "Indirect";
                }
            }
        }
    }
    return(Transition);
}
//Other cases should follow similarly

```



```

long double Layer::k_zo(Layer barrier, Carrier &carrier, long double Voltage, long double kd)
{
    long double ns, F, E_0, E_1, E_2;
    fstream file;

    //Set E_0_Test <> 0 to check E_0, E_1, and E_2 variation
    int E_0_Test=0;
    if (E_0_Test){
        Voltage = 0.;
        carrier.ListContent();
        file.open("E_0_Test.dat", ios::out);
        file<<"Voltage\tSpaceCharge\tField\tGroundStateEnergy\n";
        do{

            ns = (static_index*barrier._thickness() + barrier.static_index*_thickness())
                *eo*(hBar*kd/(-e*barrier._thickness()*carrier.effective_mass*carrier.mobility))
                - (Voltage*barrier.static_index*eo/(-e*barrier._thickness()));
            F = ns*(-e)/(static_index*eo);

            //Ground and the first two excited states
            E_0 = pow(0.5*hBar*hBar/carrier.effective_mass, 1./3.)*
                pow(pow(1.5*p*e*F*0.7587, 2.), 1./3.);
            E_1 = pow(0.5*hBar*hBar/carrier.effective_mass, 1./3.)*
                pow(pow(1.5*p*e*F*1.7540, 2.), 1./3.);
            E_2 = pow(0.5*hBar*hBar/carrier.effective_mass, 1./3.)*
                pow(pow(1.5*p*e*F*2.7575, 2.), 1./3.);
            file << Voltage<<"\t"<<ns<<"\t"<<F<<"\t"
                << E_0<<"\t"<<E_1<<"\t"<<E_2<<endl;
            Voltage += 0.001;

        } while (Voltage <= 10.5);

        file.close();
        exit(0);
    }
    //E_0 test ends

    //Calculating the space charge as a function of Reverse Bias
    ns = -(static_index*barrier._thickness() + barrier.static_index*_thickness()) +
        barrier.static_index*Voltage ;
    ns = ns * eo /(e*barrier._thickness());

    //F is the electric field in the space charge region (assumed constant)
    //F was derived by solving the poissons equation where ns is the 2-D space charge density

    F = ns*e/(static_index*eo);

    //E_0 is the ground state energy of the triangular well.
    //0.7587 comes from correcting 0.75 as in Ando's 2-D distributions

```

```

E_0 = pow(0.5*hBar*hBar/carrier.effective_mass, 1./3.)*pow(pow(1.5*p*e*F*0.7587, 2.), 1./3.);

//Return k_zo
//Or in otherwards the barrier lowering by the accumulation layers effectively
//increases the carrier drift by k_zo

return pow(2*carrier.effective_mass*E_0, 0.5)/hBar;
}

```

```

//Definition of the Device class.

#ifndef DEVICE_H
#define DEVICE_H

#include <string.h>
#include "Layer.h"
#include "barrier.h"

struct Dimension{
    long double electrical;
    long double optical;
};

class Device{
public:
    Layer top_contact;
    Layer drift;
    Barrier barrier;
    Layer bottom_contact;
    Layer substrate;
    Dimension area;
    long double bias;
    long double temperature;
    int Set_Temperature(long double t);
    long double ThermalDarkCurrent (long double alpha);
};

inline int Device::Set_Temperature(long double t)
{
    if (t<=1000){
        temperature = t;
        top_contact.temperature = t;
        drift.temperature = t;
        barrier.temperature = t;
        bottom_contact.temperature = t;
        substrate.temperature = t;

        return 1;
    }else
        return 0;
}

#endif

```

```

//Entry point to the switching program

#include <stdlib.h>
#include <iostream>
#include <string>
#include <fstream>
#include <conio.h>
#include <complex>
#include <iomanip>
#include "bessel.h"
#include "time.h"
#include "date.h"
#include "Constants.h"
#include "Carrier.h"
#include "Layer.h"
#include "Read.h"
#include "Device.h"
#include "Barrier.h"

//These are for GaAs
long double F0=5.95e5, TD=417., w0=5.46895e13;
time_of_day now;
date today;

void PlotRHS(Device &device, Carrier &carrier, long double kd, const long double jz_tunneling)
{
    long double k0, k0p, T, Te;
    long double Right_of_eqn_5, DriftField;
    long double x, xe;
    long double alpha;
    string FileName;
    fstream File;

    //Initializing the values for Iteration
    T          = device.drift.temperature;
    Te         = carrier.temperature;
    x          = hBar*w0/(k*T);
    alpha      = e*device.drift._doping_density()*pow(2*k*TD/
    (p*carrier.effective_mass), 0.5)*F0/(exp(x)-1);
    DriftField = hBar*kd/(carrier.effective_mass*carrier.mobility);

    cout << "\n\tEnter the file Name:\t";
    cin >> FileName;

```

```

if(FileName != "n") {
    File.open(FileName.c_str(), ios::out);
    File<<"T = "<<T<<endl;
    File<<"Te\tRHS"<<endl;

    long double Ti, TR, Step, TL;

    Ti = T;

    cout<<"\n\tEnter lower Limit: ";
    cin>>TL;
    cout<<"\n\tEnter Upper Limit: ";
    cin>>TR;
    cout<<"\n\tEnter Step size: ";
    cin>>Step;

    while(Ti<=TR) {
        xe = hBar*w0/(k*Ti);
        bessik01a(xe/2., k0, k0p);
        Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
        File<<setw(12)<<setprecision(8);
        File<<Ti<<"\t"<<Right_of_eqn_5<<endl;
        Ti += Step;
    }
    File.unsetf();
    File.close();
    cout<<"\n\tFile Done";
}

cout << "\n\tj_z = "<<jz_tunneling<<endl;
cout << "\n\tCarrier Temperature? ";
cin >> Te;
carrier.temperature = Te;
}

```

```

int Get_Te_BiSectionMethod
(Device &device, Carrier &carrier, long double kd, const long double jz_tunneling)
{
    long double k0, k0p, T, Te;
    long double Right_of_eqn_5, DriftField;
    long double f;
    long double x, xe;
    long double alpha;

```

```

cout<<"\n\tCurrent input to the Bisection Method equation\t"<<jz_tunneling<<endl;

//Initializing the values for Iteration
T           =      device.drift.temperature;
Te          =      carrier.temperature;
x           =      hBar*w0/(k*T);
alpha       =      1.6e-10*device.drift._doping_density()*
                  pow(2*k*TD/(p*carrier.effective_mass), 0.5)*F0/(exp(x)-1);

DriftField  =      hBar*kd/(carrier.effective_mass*carrier.mobility);

//Setting the limits for Bisection methods
int i(0), count(0);
int N(1000);
long double a(T), b(10000);
long double Tolerance(1.e-300);

//Exit if Lattice tempertaure Falls out of [a,b]
if (Te<a || Te>=b) {
    cout <<"\n\tImproper Bisection limits found"<<endl;
    cout<<"\tLimits are a= "<<a<<" and b= "<<b<<endl;
    cout<<"\tDevice temperature, T_e = "<<Te<<endl;
    exit(0);
}
//Else Proceed to iterate
while (i<N) {
    Te = a + (b-a)/2.;
    xe = hBar*w0/(k*Te);
    bessik01a(xe/2., k0, k0p);
    Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
    f = -jz_tunneling*DriftField + Right_of_eqn_5;

    if (f ==0. || (b-a)/2. < Tolerance || count>=20) {

        cout<<"\n\t(b-a)/2.\t"<<(b-a)/2.<<endl;
        cout<<"\n\tTolerance\t"<<Tolerance<<endl;

        return 1;
    }
    i = i+1;

    long double f_a, f_b, f_p, f_p_Old;

    f_p = f;

    xe = hBar*w0/(k*a);
    bessik01a(xe/2., k0, k0p);
    Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;

```

```

    f_a = -jz_tunneling*DriftField + Right_of_eqn_5;

    xe = hBar*w0/(k*b);
    bessik01a(xe/2., k0, k0p);
    Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
    f_b = -jz_tunneling*DriftField + Right_of_eqn_5;

    if (i==0)
        f_p_Old = f_p;
    else {
        if (f_p_Old == f_p)
            count++;
        else
            f_p_Old = f_p;
    };

    if (f_a*f_p > 0.)
        a = Te;
    else
        b = Te;
}

cout<<"\nMethod failed after "<<N<<" iterations"<<endl;
return 0.;
}

int Get_Te_BiSectionMethod
(Device &device, Carrier &carrier, long double kd, const long double jz_tunneling)
{
    long double k0, k0p, T, Te;
    long double Right_of_eqn_5, DriftField;
    long double f;
    long double x, xe;
    long double alpha;

    //Initializing the values for Iteration
    T          = device.drift.temperature;
    Te         = carrier.temperature;
    x          = hBar*w0/(k*T);
    alpha      = e*device.drift._doping_density()*pow(2*k*TD/
        (p*carrier.effective_mass), 0.5)*F0/(exp(x)-1);
    DriftField = hBar*kd/(carrier.effective_mass*carrier.mobility);

    //Setting the limits for Bisection methods
    int i(0), count(0);
    int N(300);

```

```

long double a(0.1), b(10);
long double Tolerance(1e-30);

xe = hBar*w0/(k*Te);
cout<<"\n\txe:\t"<<xe<<endl;

//Exit if Lattice tempertaure Falls out of [a,b]
if (xe<a || xe>=b) {
    cout<<"\nxe:\t"<<xe<<endl;
    cout<<"\n\tImproper Bisection limits found"<<endl;
    cout<<"\tLimits are a= "<<a<<" and b= "<<b<<endl;
    cout<<"\tDevice temperature, T_e = "<<Te<<endl;
    exit(0);
}
//Else Proceed to iterate
while (i<N) {
    xe = a + (b-a)/2.;
    //xe = hBar*w0/(k*Te);
    bessik01a(xe/2., k0, k0p);
    Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
    f = -jz_tunneling*DriftField + Right_of_eqn_5;

    if (f==0. || (b-a)/2. < Tolerance || count>=20) {
        if ((b-a)/2. < Tolerance) {
            cout<<"Tolerance met\t"<<endl;
            cout<<"Uncertianity is\t"<<(b-a)/2.<<endl;
        }

        if (count>=20) {
            cout<<"Converging of Te met"<<endl;
        }

        if (f==0.)
            cout<<"Exact solution found"<<endl;

        carrier.temperature = hBar*w0/(xe*k);
        cout<<"\n\tThe Root is = "<<Te;
        return 1;
    }

    i = i+1;

    long double f_a, f_b, f_p, f_p_Old;

    f_p = f;

    xe = a;
    bessik01a(xe/2., k0, k0p);
    Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;

```



```

f_a = -jz_tunneling*DriftField + Right_of_eqn_5;

xe = b;
bessik01a(xe/2., k0, k0p);
Right_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
f_b = -jz_tunneling*DriftField + Right_of_eqn_5;

if (i==0)
    f_p_Old = f_p;
else {
    if (f_p_Old == f_p)
        count++;
    else
        f_p_Old = f_p;
};

if (f_a*f_p > 0.)
    a = xe;
else
    b = xe;

cout << "jz_tunneling:\t"          << jz_tunneling << endl;
cout << "Iteration ["              << i << "]: " << endl;
cout << "Te:\t"                    << hBar*w0/(xe*k) << endl;
cout << "Right_of_eqn_5:\t"        << Right_of_eqn_5 << endl;
cout << "jz_tunneling*DriftField:\t" << jz_tunneling*DriftField << endl;
cout << "\n\tcount\t"              << count << endl;
}

cout << "\nMethod failed after " << N << " iterations" << endl;
return 0.;
}

```

```

long double Get_Te_NewtonRaphson
(Device device, Carrier &carrier, long double kd, const long double jz_tunneling)
{
    long double k0, k0p, Te, Te0;
    long double Left_of_eqn_5, DriftField, Tolerance=0.1;
    long double f, fP, alpha;
    long double x, xe;
    int i = 1, N0 = 200;

    x = hBar*w0/(k*device.drift.temperature);

    alpha = e*device.drift.doping_density()*pow(2*k*TD/
        (p*carrier.effective_mass), 0.5)*F0/(exp(x)-1);
}

```

```

DriftField = hBar*kd/(carrier.effective_mass*carrier.mobility);
Te0 = carrier.temperature;

bool PlotEq = true;
if(PlotEq) {
    cout<<"Hello"<<device.drift.temperature<<endl;
    Te0 = 4.2;
    long double Te0_Res = 0.1;
    string EqPlotName("Right of Eq for 50 K.dat");

    fstream EqFile;
    EqFile.open(EqPlotName.c_str(), ios::out);
    do {
        xe = hBar*w0/(k*Te0);
        bessik01a(xe/2., k0, k0p);
        cout<<"k0\t"<<k0<<endl;
        Left_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
        EqFile<<Te0<<" "<<Left_of_eqn_5/DriftField<<endl;
        Te0 += Te0_Res;
    } while (Te0<1000);
    exit(0);
}

while (i<=N0) {
    xe = hBar*w0/(k*Te0);
    bessik01a(xe/2., k0, k0p);
    Left_of_eqn_5 = alpha*(exp(x-xe)-1)*pow(xe, 0.5)*exp(xe/2.)*k0;
    f      =      -jz_tunneling*DriftField + Left_of_eqn_5;
    fP     =      alpha*(xe, 0.5)*exp(xe/2.) * ( 0.5*(exp(x-xe)-1)*k0*(1.+1./xe) +
        (exp(x-xe)-1.)*k0p - k0*exp(x-xe) );
    Te     =      Te0 - f/fP;

    if (fabs(Te-Te0)<Tolerance) {
        carrier.temperature = Te;
        cout<<"\n\tThe Root is = "<<Te;
        return 1;
    }
    i+=1;
    Te0 = Te;
}
cout<<"\nRoot finding for Te failed after "<<N0<<" iterations"<<endl;
return 0.;
}

```

```

void main()
{
    Device device;
    fstream f;
    string Buff;
    long double kd, kzo, jz_in, jz_out;
    long double old_kd, new_kd;
    int count = 0, quit;
    long double VoltageStep = 0.1;

    //Defining the structure
    device.top_contact      = Layer("film", "GaAs", 200e-9, 'n', 1e24);
    device.drift            = Layer("film", "GaAs", 90e-9, 'n', 5e23);
    device.barrier          = Barrier("film", "GaAs", 500e-9, 'n', 1e2, "Al", 0.45);
    device.bottom_contact  = Layer("film", "GaAs", 700e-9, 'n', 1e24);
    device.substrate       = Layer("bulk", "GaAs", 400e-6, 'n', 1e20);

    device.area.electrical  = 400e-6*400e-6 ;      //in micron
    device.area.optical     = 260e-6*260e-6 ;      //in micron

    long double temperature;
    string FName;
    fstream file;

    cout << "\n\tEnter the Lattice Temperature";
    cin >> temperature;
    device.Set_Temperature(temperature);

    //Do not call before Device constructor and TemperatureSet routines.
    device.barrier.PlotThermalCurrent
        (device.drift, device.area.electrical, "ThermalAt77K.dat", 0.1, 5.0, 0.01);

    //Opening and File Headings
    cout << "\n\tGenerates a Current vs. Voltage for "<<temperature<<" K";
    cout << "Enter FileName";
    cin >> FName;
    file.open(FName.c_str(), ios::out);

    //Set parameters for switching
    Carrier carrier(device.drift);
    carrier.ListContent();

    file << "Device temperature ="<<device.temperature<<endl;
    file << "Bias (V)\tKd(m^-1)\tTe\tCurr.Density(A/m^2)"<<endl;

    long double barrier = device.barrier.SetBandOffset(device.drift);
    long double mW = carrier.effective_mass;
    long double mB = carrier.set_effective_mass(device.barrier);
    long double W1 = device.drift._thickness();

```

```

ofstream out;
out.open("Temp77K1e17-3.txt", ios::out);
out << "old_kd\tnew_kd\t(new_kd-old_kd)\tcar.temp"<<endl;

//According to design the device should switch only for the reverse bias.
//But In this program the bias value should be positive (i.e Reverse Bias)
//Since the - sign of the Voltage has
//been already taken care of in the
//Poisson Equations

device.bias = 1.2; //Starting value for the Bias

device.barrier.GetTunnelingCurrent("Trapezoidal", "WKB", device.drift, 2.e5, 1.e3);

do {
    //Seeds for a give Bias voltage
    //Carrier Temperature is set to Lattice temperature
    carrier.temperature = device.drift.temperature;
    //Carrier Drift is set to zero.
    kd = 0.;

    out<<"Drift Layer Temperature = "<<device.drift.temperature<<" K"<<endl;
    out<<"Device Bias = "<<device.bias<<" V"<<endl<<endl;

    //iteration loop
    quit = 0;
    count = 0;
    do {
        old_kd = kd;

        //kzo give the enery increase of carriers resulted from the Barrier lowering
        //and is a function of the Voltage.
        //kzo = minimum for V = 0.
        //Voltage is the only variable parameter that determines kzo
        kzo = device.drift.k_zo(device.barrier, carrier, device.bias, kd);

        //Setting the Drift velocity of Carrier
        //Drift velocity will be increased by the contribution (k_zo) from Barrier
        lowering

        carrier.SetDrift(0., 0., kd+kzo);
        cout<<"\n\tBias Voltage"<<device.bias;
        jz_out = device.barrier.GetTunnelingCurrent (carrier, "Trapezoidal",
                                                    Gundlach", barrier, mW, mB, device.bias, W1);

        //-----Finding the starting point for the iteration!-----

```

```

if (jz_out<1.e-30) {
    cout<<"\nPlease wait while a starting point is found..."<<endl;
    cout<<"\n\tBias Voltage\tjz_out"<<endl;
    while (jz_out<1.e-30) {
        device.bias += VoltageStep;
        cout<<"\t"<<device.bias<<"\t"<<jz_out<<endl;
        kzo = device.drift.k_zo(device.barrier, carrier,
                                device.bias, kd);
        carrier.SetDrift(0., 0., kd+kzo);
        jz_out = device.barrier.GetTunnelingCurrent
            (carrier, "Trapezoidal", "WKB", barrier, mW, mB,
             device.bias, W1);
    }
    cout<<"The starting bias is set to "<<device.bias<<" V"<<endl;
}

//Carrier momentum(kd) from
kd = carrier.effective_mass*jz_out/(e*device.drift._doping_density()*hBar);
new_kd = kd;
jz_in = jz_out;
Get_Te_BiSectionMethod(device, carrier, kd, jz_in);
cout<<"\n\tNew kd = "<<kd;
cout<<"\n\tNew Te = "<<carrier.temperature;
quit++;

out <<old_kd<<"\t"<<new_kd<<"\t"<<new_kd-old_kd<<"\t"
    <<carrier.temperature<<endl;

if (new_kd<old_kd) {
    cout<<"\n\tReverse Error encountered at bias voltage "<<device.bias;
    exit(0);
}

if (new_kd-old_kd<=0.001) {
    count++;
    cout<<"\n\tnew_kd-old_kd\t"<<new_kd-old_kd<<endl;
    cout<<"\tcount\t"<<count<<endl;
}

} while (count<30);

```

```

        if (count==30) {
            cout    <<"device.bias"<<"\t"<<"kd"<<"\t"<<"carrier.temperature"
                    <<"\t"<<"jz_out"<<endl;

            cout    <<device.bias<<"\t"<<kd<<"\t"<<carrier.temperature
                    <<"\t"<<jz_out<<endl;
            file     <<device.bias<<"\t"<<kd<<"\t"<<carrier.temperature<<"\t"
                    <<jz_out<<endl;
        }

        device.bias += VoltageStep;

    } while (device.bias<20 && quit<1000);

    out.close();
    file.close();
    cout<<"\n End of Program";
}

```

Bibliography

- ¹ Michael W. Werner, “The Atlas SIRTf”, *Infrared Phys. Technol.*, vol. 35, p. 539, 1994.
- ² P. Garcia-Lario, Arturo Manchado, Ana Ulla, and Minia Manteiga, “Infrared Space Observatory Observations of IRAS 16594-4656: A New Proto-Planetary Nebula with a Strong 21 Micron Dust Feature”, *Astrophys. J.*, vol. 513, p. 941, 1999.
- ³ H. Izumura, O. Hashimoto, K. Kawara, I. Yamamura, L. B. F. M. Waters, “A detached dust shell surrounding the J-type carbon star Y Canum Venaticorum”, *Astronomy and Astrophysics*, vol. 315, p. L221, 1996.
- ⁴ B. Bezard, D. Gautier, and A. Marten, “Detectability of HD and non-equilibrium species in the upper atmospheres of the gas giant planets from their submillimeter spectrum”, *Astron. & Astrophys.*, vol. 161, p. 387, 1986.
- ⁵ E. E. Haller, “Advanced Far-Infrared Detectors”, *Infrared Phys.*, vol. 35, p. 127, 1994.
- ⁶ D. M. Watson, M. T. Gupitill, J. E. Huffman, T. N. Krabach, S. N. Raines, and S. Satyapal, “Germanium blocked-impurity-band detector arrays: Unpassivated devices with bulk substrates”, *J. Appl. Phys.*, vol. 74, p. 4199, 1993.
- ⁷ J. E. Huffman, A. G. Crouse, B. L. Halleck, T. V. Downes, and T. L. Herter, “Si:Sb blocked impurity band detectors for infrared astronomy”, *J. Appl. Phys.*, vol. 72, p. 273, 1992.
- ⁸ L. A. Reichertz, B. L. Cardozo, J. W. Beeman, D. I. Larsen, S. Tschanz, G. Jakob, R. Katterloher, N. M. Haegel, and E. E. Haller, *First Results on GaAs blocked impurity band (BIB) structures for far-infrared detector arrays* **5883**, 2005, 58830Q1-58830Q8.

- ⁹ S. G. Carter, V. Ciulin, M. Hanson, A. S. Huntington, C. S. Wang, A. C. Gossard, L. A. Coldren, and M. S. Sherwin, "Terahertz-optical mixing in undoped and doped GaAs quantum wells: From excitonic to electronic intersubband transitions", *Physical Review B*, vol. 72, p. 155309, 2005.
- ¹⁰ Bradley Ferguson, and Xi-Cheng Zhang, "Materials for terahertz science and technology", *Nature Mat.*, vol. 1, p. 26-33, 2002.
- ¹¹ C. Stellmach, A. Hirsch, G. Nachtwei, Yu. B. Vasilyev, N. G. Kalugin, and G. Hein, "Fast terahertz detectors with spectral tunability based on quantum hall corbino devices", *Appl. Phys. Lett.*, vol. 87, p. 133504, 2005.
- ¹² F. D. Shepherd, "Infrared internal emission detectors", *Proc. SPIE*, vol. 1735, p. 250, 1992.
- ¹³ W. F. Kosonocky, "State-of-art in Schottky-barrier IR image sensors", *SPIE*, vol. 1685, p. 2, 1992.
- ¹⁴ T.-L. Lin, A. Ksendzov, S. M. Dejweski, E. W. Jones, R. W. Fathauer, T. N. Krabach, and J. Maserjian, "SiGe/Si Heterojunction Internal Photoemission Long-Wavelength Infrared Detectors Fabricated by Molecular Beam Epitaxy", *IEEE Trans. on Electron Devices*, vol. 38, p. 1141, 1991.
- ¹⁵ B-Y. Tsaur, C. K. Chen, and S. A. Marino, "Long-wavelength $\text{Ge}_x\text{Si}_{1-x}$ /Si heterojunction infrared detectors and focal plane arrays", *Proc. SPIE*, vol. 1540, p. 580, 1991.
- ¹⁶ D. D. Coon, R. P. Devaty, A. G. U. Perera and R. E. Sherriff, "Interfacial Work Functions and Extrinsic Silicon Infrared Photocathodes", *Appl. Phys. Lett.*, vol. 55, p. 1738, 1989.
- ¹⁷ M. S. Unlu, G. Ulu, and M. Gokkavas, in *Photodetectors and Fiber Optics*, edited by H. S. Nalwa (Academic Press, NY, 2001), pp. 97-201.
- ¹⁸ S. C. Jain and D. J. Roulston, "A simple expression for band gap narrowing (BGN) in heavily doped Si, Ge, GaAs and $\text{Ge}_x\text{Si}_{1-x}$ strained layers", *Solid-St. Electron.*, vol. 34, p. 453, 1991.
- ¹⁹ M. V. Klein, T. E. Furtac, *Optics* (Wiley, New York, 1986).
- ²⁰ J. S. Blakemore, "Semiconducting and other major properties of gallium arsenide", *J. Appl. Phys.*, vol. 53, p. R123, 1982.

- ²¹ S. Adachi, *GaAs and Related Materials* (WorldScientific, Singapore, 1994).
- ²² V. E. Vickers, "Model of Schottky barrier hot-electron-mode Photodetection", *Appl. Opt.*, vol. 10, p. 2190, 1971.
- ²³ J. M. Mooney and J. Silverman, "The theory of hot-electron photoemission in Schottky-barrier IR detectors", *IEEE Trans. Electron Devices*, vol. 32, p. 33, 1985.
- ²⁴ M. J. Kane, S. Millidge, M. T. Emeny, D. Lee, D. R. P. Guy, and C. R. Whitehouse, in *Intersubband transitions in Quantum Wells*, edited by E. Rosenchenr, B. Vinter and B. Levine (Plenum, New York, 1992).
- ²⁵ S. G. Matsik, M. B. M. Rinzan, D. G. Esaev, A. G. U. Perera, H. C. Liu, and M. Buchanan, "20 μm cutoff heterojunction interfacial work function internal photoemission detectors", *Appl. Phys. Lett.*, vol. 84, p. 3435, 2004.
- ²⁶ H. C. Liu, "Noise gain and operating temperature of quantum well infrared photodetectors", *Appl. Phys. Lett.*, vol. 61, p. 2703, 1992.
- ²⁷ M. Ershov and H. C. Liu, "Low-frequency noise gain and photocurrent gain in quantum well infrared photodetectors", *J. Appl. Phys.*, vol. 86, p. 6580, 1999.
- ²⁸ A. G. U. Perera, H. X. Yuan, and M. H. Francombe, "Homojunction internal photoemission far-infrared detectors: Photoresponse performance analysis", *J. Appl. Phys.*, vol. 77, p. 915, 1995.
- ²⁹ R. Colombelli, F. Capasso, C. Gmachl, A. L. Hutchinson, D. L. Sivco, A. Tredicucci, M. C. Wanke, A. M. Sargent, and A. Y. Cho, "Far-infrared surface-plasmon quantum-cascade lasers at 21.5 μm and 24 μm wavelengths", *Appl. Phys. Lett.*, vol. 78, p. 2620, 2001.
- ³⁰ D. G. Esaev, S. G. Matsik, M. B. M. Rinzan, A. G. U. Perera, H. C. Liu, and M. Buchanan, "Resonant cavity enhancement in heterojunction GaAs/AlGaAs terahertz detectors", *J. Appl. Phys.*, vol. 93, p. 1879, 2003.
- ³¹ W. Z. Shen and A. G. U. Perera, "Photoconductivity in homojunction internal photoemission far-infrared detectors", *Infrared Physics and Technology*, vol. 39, p. 329, 1998.

- ³² W. Z. Shen, A. G. U. Perera, S. K. Gamage, H. X. Yuan, H. C. Liu, M. Buchanan and W. J. Schaff, "A spectroscopic study of GaAs homojunction internal photoemission for infrared detectors", *Infrared Physics & Technology*, vol. 38, p. 133, 1997.
- ³³ S. M. Sze, *Physics of Semiconductor Devices* (John Wiley & Sons, New York, 1981).
- ³⁴ N. F. Mott, *Metal-Insulator Transitions* (Barnes and Noble, New York, 1974).
- ³⁵ A. G. U. Perera, H. X. Yuan, S. K. Gamage, W. Z. Shen, M. H. Francombe, H. C. Liu, M. Buchanan, W. J. Schaff, "GaAs multilayer p^+ -i homojunction far-infrared detectors", *J. Appl. Phys.*, vol. 81, p. 3316, 1997.
- ³⁶ A. G. U. Perera, S. G. Matsik, M. B. M. Rinzan, A. Weerasekara, M. Alevli, H. C. Liu, M. Buchanan, B. Zvonkov, and V. Gavrilenko, "The Effects of Light-Heavy Hole Transitions on the Cutoff Wavelengths of Far Infrared Detectors", *Infrared Phys. Technol.*, vol. 44, p. 347, 2003.
- ³⁷ A. L. Korotkov, A. G. U. Perera, W. Z. Shen, J. Herfort, K. H. Ploog, W. J. Schaff, and H. C. Liu, "Free-carrier absorption in Be and C doped GaAs epilayers and far infrared detector applications", *J. Appl. Phys.*, vol. 89, p. 3295, 2001.
- ³⁸ A. G. U. Perera, S. G. Matsik, M. Ershov, Y. W. Yi, H. C. Liu, M. Buchanan, and Z. R. Wasilewski, "Effects of traps on the dark current transients in GaAs/AlGaAs quantum-well infrared photodetectors", *Physica E*, vol. 7, p. 130, 2000.
- ³⁹ G. M. Williams, R. E. DeWames, C. W. Farley and R. J. Anderson, "Excess tunnel currents in AlGaAs/GaAs multiple quantum well infrared detectors", *Appl. Phys. Lett.*, vol. 60, p. 1324, 1992.
- ⁴⁰ S. Zangoie, M. Schubert, D. W. Thompson, and J.A. Woollam, "Infrared response of multiple-component free-carrier plasma in heavily doped p-type GaAs", *Appl. Phys. Lett.*, vol. 78, p. 937, 2001.
- ⁴¹ R. F. Kirkman, R. A. Stradling, and P. J. Lin-Chung, "An infrared study of the shallow acceptor states in GaAs", *J. Phys. C*, vol. 11, p. 419, 1978.
- ⁴² N. O. Lipari, A. Baldereschi, "Angular Momentum Theory and Localized States in Solids. Investigation of Shallow Acceptor States in Semiconductors", *Phys. Rev. Lett.*, vol. 25, p. 1660, 1970.

- ⁴³ A. G. U. Perera, *Semiconductor Photoemissive Structures for Far Infrared Detection, Handbook of Thin Film Devices Frontiers of Research, Technology and Applications*, edited by M. H. Francombe, A. G. U. Perera and H. C. Liu, Vol 2- *Semiconductor Optics* (Academic Press, NY, 2000).
- ⁴⁴ M. Seon, M. Holtz, W. M. Duncan, and T. S. Kim, "Raman studies of heavily carbon doped GaAs", *J. Appl. Phys.*, vol. 85, p. 7224, 1999.
- ⁴⁵ F. Cappaso, et al., "Quantum cascade lasers: Ultrahigh-speed operation, optical wireless communication, narrow linewidth and far-infrared emission", *IEEE J. Quan. Elec.*, vol. 38, p. 511, 2002.
- ⁴⁶ A. G. U. Perera and H. C. Liu, in *Handbook of Thin Film Devices: Semiconductor optical and Electro optical Devices*, edited by M. H. Francombe (Academic Press, , 2001).
- ⁴⁷ Y. N. Yang, D. D. Coon and P. F. Shepard, "Thermionic Emission in Silicon at Temperatures below 30K", *Appl. Phys. Lett.*, vol. 752, p. 752, 1984.
- ⁴⁸ A. Shen, H. C. Liu, M. Gao, E. Dupont, M. Buchanan, J. Ehret, G. J. Brown, and F. Szmulowicz, "Resonant-cavity-enhanced p-type GaAs/AlGaAs quantum-well infrared photodetectors", *Appl. Phys. Lett.*, vol. 77, p. 2400, 2000.
- ⁴⁹ S. G. Matsik, M. B. M. Rinzan, A. G. U. Perera, H. C. Liu, Z. R. Wasilewski, and M. Buchanan, "Cutoff Tailorability of Heterojunction Terahertz Detectors", *Appl. Phys. Lett.*, vol. 82, p. 139, 2003.
- ⁵⁰ A. G. U. Perera, S. G. Matsik, B. Yaldiz, H. C. Liu, A. Shen, M. Gao, Z. R. Wasilewski, and M. Buchanan, "Heterojunction wavelength tailorable far infrared photodetectors with response out to 70 μm ", *Appl. Phys. Lett.*, vol. 78, p. 2241, 2001.
- ⁵¹ H. X. Yuan and A. G. U. Perera, "Space-charge-limited conduction in Si $n^+ - i - n^+$ homojunction far-infrared detectors", *J. Appl. Phys.*, vol. 79, p. 4418, 1996.
- ⁵² M. L. Huberman, A. Ksendzov, A. Larsson, R. Terhune, and J. Maserjian, "Optical Absorption by free holes in heavily doped GaAs", *Phys. Rev. B*, vol. 44, p. 1128, 1991.
- ⁵³ D. J. Lockwood, and Z. R. Wasilewski, "Please enter the title", *Phys. Rev. B*, vol. 70, p. 155202, 2004.

- ⁵⁴ M. B. M. Rinzan, D. G. Esaev, A. G. U. Perera, S. G. Matsik, G. Von Winckel, A. Stintz, and S. Krishna, “Free carrier absorption in Be doped epitaxial AlGaAs thin films”, *Appl. Phys. Lett.*, vol. 85, p. 5236, 2004.
- ⁵⁵ H.C. Liu, C.Y. Song, A.J. Spring Thorpe, and J.C. Cao, “Terahertz quantum-well photodetector”, *Appl. Phys. Lett.*, vol. 84, p. 4068, 2004.
- ⁵⁶ Jian-Qiang Lü, Michael S. Shur, Jeffrey L. Hesler, Liangquan Sun, and Robert Weikle, “Terahertz detector utilizing two-dimensional electronic fluid”, *IEEE Electron Device Lett.*, vol. 19, p. 373, 1998.
- ⁵⁷ D. G. Esaev, M. B. M. Rinzan, S. G. Matsik, and A. G. U. Perera, “Design and optimization of GaAs/AlGaAs heterojunction infrared detectors”, *J. Appl. Phys.*, vol. 96, p. 4588, 2004.
- ⁵⁸ G. Borghs, K. Bhattacharyya, K. Deneffe, P. Van Mieghem, and R. Mertens, “Band-gap narrowing in highly doped n - and p -type GaAs studied by photoluminescence spectroscopy”, *J. Appl. Phys.*, vol. 66, p. 4381, 1989.
- ⁵⁹ A. Van der Ziel, *Noise in Solid State devices and Circuits* (Wiley-Interscience, New York, 1986).
- ⁶⁰ R. Mosca, P. Bussei, S. Franchi, P. Frigeri, E. Gombia, A. Carnera, and M. Peroni, “Terahertz detector utilizing two-dimensional electronic fluid”, *J. Appl. Phys.*, vol. 93, p. 9709, 2003.
- ⁶¹ D. D. Coon, S. N. Ma and A. G. U. Perera, “Farey-fraction Frequency Modulation in the Neuronlike output of Silicon p - i - n diodes at 4.2K”, *Phys. Rev. Lett.*, vol. 58, p. 1139, 1987.
- ⁶² D. D. Coon and A. G. U. Perera, “Spiketrain Generation and Intensity to Frequency Conversion”, *Appl. Phys. Lett.*, vol. 55, p. 478, 1989.
- ⁶³ A. G. U. Perera, S. R. Betarbet and M. H. Francombe, “Multispectral Transient Sensing based on Retinal concepts in a Parallel Processor”, in *World Congress on Neural Networks 1993*, Lawrence Erlbaum Associates, NJ, 1993, pp. IV 835IV 838.
- ⁶⁴ K. Hess, T. K. Higman, M. A. Emanuel, and J. J. Coleman, “New ultrafast switching mechanism in semiconductor heterostructures”, *J. Appl. Phys.*, vol. 60, p. 3775, 1986.

- ⁶⁵ A. Weerasekera, S. G. Matsik, G. Cymbaluyk, and A. G. U. Perera, "Grouping behavior of inter-pulse time intervals for triggered pulses in an AlGaAs/InGaAs multilayer structure", *Physica D*, vol. 215, p. 159, 2006.
- ⁶⁶ A. G. U. Perera, S. G. Matsik, V. Y. Letov, H. C. Liu, M. Gao, M. Buchanan, W. J. Schaff, "Spontaneous and Triggered Pulsing in GaAs/InGaAs Multi-quantum Well Structures", *Solid State Elec.*, vol. 45, p. 1121, 2001.
- ⁶⁷ A. G. U. Perera, S. G. Matsik and S. R. Betarbet, "Nonlinear Dynamics in a Parallel Neural Network Processor" in *Visual Information Processing IV, Proc. SPIE*, vol. 2488, p. 363, 1995.
- ⁶⁸ A. Wacker and E. Schöll, "Oscillatory instability in the heterostructure hot electron diode", *Appl. Phys. Lett.*, vol. 59, p. 1702, 1991.
- ⁶⁹ C. Song and K. P. Roenker, "S-type switching characteristics from transverse transport in multi-quantum well diodes", *J. Appl. Phys.*, vol. 72, p. 4417, 1992.
- ⁷⁰ K. Hess and G. J. Iafrate, in *Heterojunction Band Discontinuities*, edited by F. Capasso and G. Margaritondo (North Holland, New York, 1987).
- ⁷¹ E. L. Murphy and R. H. Good, "Thermionic Emission, Field Emission, and the Transition Region", *Physical Review*, vol. 102, p. 1464, 1956.
- ⁷² T. K. Higman, J. M. Higman, M. A. Emanuel, K. Hess, and J. J. Coleman, "Theoretical and experimental analysis of the switching mechanism in heterostructure hot-electron diodes", *J. Appl. Phys.*, vol. 62, p. 1495, 1987.
- ⁷³ K. H. Gundlach, "... resonant Fowler-Nordheim tunneling ...", *Solid-State Electronics*, vol. 9, p. 949, 1966.
- ⁷⁴ E. M. Conwell and M. O. Vassell, "High Field Transport in *n*-Type GaAs", *Physical Review*, vol. 166, p. 797, 1968.

Acronyms

HEIWIP heterojunction interfacial workfunction internal photoemission

HIWIP homojunction interfacial workfunction internal photoemission

RCA resonant cavity architecture

BLIP background limited infrared photodetector

IWIP interfacial workfunction internal photoemission

OMCVD organometallic chemical vapor deposition

MBE molecular beam epitaxy

SI semi-insulating

FOV field of view

Be berillium

Si silicon

Ga gallium

Ge germanium

C carbon

BIB blocked-impurity-band

FPA focal plane array

THz Terahertz

FIR far infrared

SIMS secondary ion mass spectroscopy

λ_0 threshold wavelength

Reststrahlen band - A region in the infrared that is highly phonon active and absorbs most of the infrared radiation

QWIP quantum well infrared photodetector

MCT mercury cadmium telluride

NDR negative differential resistance

2-DEG two dimensional electron gas

HHED hetero-structure hot electron device